UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

ANÁLISIS Y VISUALIZACIÓN DEL IMPACTO DE NOTICIAS EN REDES SOCIALES

Autor: Pablo Hergueta Ximénez

Tutor: Francisco Jurado Monroy

JULIO 2020

A mis padres…

Todo lo que tenemos
es para darlo.

Resumen

Las redes sociales, a pesar de no ser consideradas como una de las fuentes de información más fiables, siguen siendo uno de los medios más utilizados para seguir las noticias y el medio en el que se propagan más rápidamente. Sin embargo, el mundo de la escucha social está dominado primordialmente por herramientas propietarias cuyo objetivo es permitir a las corporaciones tomar decisiones empresariales más inteligentes para conseguir que crezca su negocio. Este proyecto presenta una alternativa que ofrece, aun con limitaciones, una funcionalidad similar a la de estas herramientas, pero cuyo objetivo es conocer el alcance que puede llegar a tener una noticia, sea falsa o no, dentro de las redes sociales, en concreto, dentro de la red social Twitter. Para cumplir con este objetivo se ha desarrollado una aplicación que permite, a través de una consulta introducida por el usuario, recopilar un conjunto de tuits que cumplen con los criterios solicitados, analizarlos utilizando distintas métricas y visualizar los resultados de este análisis en un tablero compuesto por distintas gráficas que se corresponden con cada una de las métricas. Finalmente, se han obtenido resultados notablemente positivos tras realizar pruebas con usuarios reales para evaluar su grado de satisfacción con respecto la utilidad y la apariencia gráfica de la herramienta.

Palabras clave: noticias, red social, escucha social, análisis y visualización, Twitter, Python, Dash

Abstract

Even though social media is not considered one of the most reliable sources of information, it continues to be one of the most used platforms to follow the news and the place where they travel faster. However, the world of social listening is primarily dominated by proprietary software tools whose goal is to allow corporations to make smarter business decisions; therefore allowing them to grow. This project presents an alternative that offers, even with limitations, similar functionality to these tools, but with a different objective: to find out the impact of a given piece of news, fake or not, on social media, specifically on Twitter. To achieve this goal, the app developed for this project allows through a query entered by the user, to collect a set of tweets that meet the requested criteria, analyze them using different metrics and view the results of this analysis on a table composed of different graphs that correspond to each of the metrics. Finally, remarkably positive results have been obtained after carrying out tests with real users to assess their degree of satisfaction with respect to the utility and graphic appearance of the tool.

Key words: news, social media, social listening, analysis and visualization, Twitter, Python, Dash

Agradecimientos

En primer lugar, quiero agradecer a mi tutor, Francisco Jurado Monroy, la atención que he recibido a lo largo de todo el proyecto y la confianza que ha depositado en mí y en este trabajo desde el comienzo, cuando era tan solo una idea. Quiero agradecer especialmente el trato recibido durante las últimas etapas del proyecto, ya que no habría sido posible llevarlo a cabo sin su disponibilidad en estas fechas que, por unas razones u otras, no han sido fáciles para todos. En segundo lugar, quiero agradecer el apoyo de mis amigos porque ellos han hecho más llevaderos los momentos difíciles y están siempre disponibles para ayudar. Por último, y no por ello menos importante, quiero agradecer especialmente el apoyo que he recibido de mis padres y de mi hermana María; gracias a la confianza que han depositado en mí estos meses, la educación que he recibido durante toda mi vida y su cariño, he llegado a ser la persona que soy ahora y he conseguido llegar hasta donde estoy.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Desde que comenzó la transición digital en los medios de comunicación, la velocidad a la que procesamos las noticias, y la información en general, ha aumentado notablemente. La democratización de la tecnología ha influido en el acceso a esta información, haciendo que sea mucho más sencillo e inmediato saber qué está sucediendo a nuestro alrededor en todo momento. Sin embargo, como se ha demostrado en los últimos años, esta inmediatez no es siempre una ventaja.

Debido a esta transformación digital, el paradigma de la prensa actual ha virado hacia un modelo en el que existen noticias impulsadas por algoritmos cuya función es maximizar el tráfico de la información y los beneficios económicos; que proceden principalmente de anuncios. De esta manera, es habitual que, en ocasiones, prevalezca la inmediatez sobre la calidad de estas noticias y, por lo tanto, es natural que las redes sociales se hayan convertido en uno de los canales de distribución principales. Esto tiene como consecuencia que proliferen y aparezcan cada vez más noticias falsas, ya sea por falta de contraste de información o de manera intencionada. Sin embargo, a pesar del poco esperanzador futuro que tienen los medios tradicionales, como la prensa impresa o la radio, se siguen considerando como medios más fiables para obtener información que las redes sociales.

Conociendo de esta situación, y la desinformación que puede llegar a existir, y sabiendo que, aun así, dos tercios de los consumidores siguen accediendo a la información a través de este medio [1], la motivación principal para el desarrollo de este proyecto ha sido poder hacer un seguimiento de manera objetiva del verdadero alcance e impacto que puede llegar a tener una noticia, sea verdadera o falsa, en las redes sociales.

1.2. Objetivos

Los objetivos principales que se han buscado conseguir al desarrollar este proyecto han sido tres:

- El desarrollo de una aplicación que permitiera hacer un análisis lo más objetivo posible, utilizando distintas métricas, de un conjunto de mensajes enviados a través de la red social Twitter.
- La implementación de un panel que permitiera visualizar, de manera detallada y dinámica, los resultados de este análisis.
- Hacer una evaluación de la satisfacción de los usuarios que utilicen esta aplicación.

Teniendo en cuenta la información proporcionada por Twitter para cada tuit [2, 3, 4], se han utilizado diversas métricas para cumplir con los objetivos de análisis y visualización planteados anteriormente. Se podrán visualizar, por ejemplo, las interacciones a lo largo del tiempo, las palabras, *hashtags*, menciones o URLs más frecuentes, la localización de los usuarios que han enviado los tuits, la distribución de las fuentes desde las que se han publicado todos los mensajes, un análisis de sentimientos y emociones o una tabla con información general que permitirá ordenar, filtrar y consultar todos los mensajes enviados a través de la red social.

1.3. Organización de la memoria

Este documento está dividido en 6 capítulos. En el segundo capítulo se hablará del estado del arte recopilando algunos ejemplos de herramientas ya existentes y similares a la desarrollada. A continuación, se recogerá el análisis previo al desarrollo del proyecto, incluyendo requisitos funcionales y no funcionales, tecnologías utilizadas, etc. Posteriormente, se pasará a detallar el diseño y el desarrollo de la aplicación, argumentando las decisiones tomadas en el proceso. En el quinto capítulo se expondrán las pruebas y los ejemplos utilizados para la obtención de resultados. Finalmente, se presentarán las conclusiones finales, el trabajo futuro y las referencias bibliográficas.

Capítulo 2

Estado del arte

En este capítulo se analizarán algunas de las diferentes herramientas más conocidas que permiten analizar y monitorizar la actividad en las redes sociales actualmente.

2.1. Hootsuite

Hootsuite es una de las herramientas más completas y el software más popular para monitorizar la actividad en redes sociales [5]. En principio, es un paquete que, con el objetivo de hacer una gestión completa de la presencia de una marca en este medio, incluye otras funcionalidades como publicar, interactuar o anunciarse, además de poder obtener informes periódicos. Además, hace posible la integración de aplicaciones y funcionalidades de terceros, de que manera que pueda completar aun más el análisis utilizando un mayor número de recursos. Sin embargo, la que más ha influido en el desarrollo de este proyecto es Hootsuite Insights.

A partir de ciertas palabras clave (marca, producto, etc.), Hootsuite Insights permite visualizar en un tablero, a través de distintos paneles configurables por el usuario, diversas piezas de información en tiempo real sobre la actividad que está teniendo lugar en diferentes redes sociales. Algunas de estas métricas son el número de menciones totales, el sentimiento, además de la evolución de estas a lo largo del tiempo, el género, los dominios más compartidos o los usuarios, los hashtags y los idiomas más populares [6].

2.2. Brandwatch

Esta solución divide su oferta en diversas plataformas, todas ellas relacionadas con el análisis de actividad en redes sociales, pero con distintos enfoques. Estos productos están conectados y forman un conjunto de soluciones que pueden escalar con las necesidades del cliente [7].

Consumer Research permite tener un acceso instantáneo a información pasada y presente utilizando motores de búsqueda propios. Puede tener como referencia canales propios, pero se puede personalizar la búsqueda para utilizar otras palabras clave o *hashtags* que puedan ser relevantes. Audiences permite analizar las audiencias estudiando sus intereses para poder encontrar público relevante, además de poder aprovechar la influencia de los usuarios más destacados. Qriously permite reemplazar anuncios publicitarios por encuestas para recopilar datos adicionales y visualizar los resultados en tiempo real. Finalmente, Vizia, completa el paquete y sirve, en mayor medida que el resto de soluciones, como inspiración para este proyecto. Permite visualizar en un panel configurable, como se puede observar en la Figura 2.1, toda la información que se quiera extraer, incluyendo la posibilidad de utilizar datos de terceros, para luego compartirla a través de pantallas con personas en todo el mundo.

FIGURA 2.1: Selección de paneles en Brandwatch Vizia. Imagen extraída de [8].

Además, el potencial y los recursos que ofrecen todas estas funcionalidades hacen que Brandwatch sea frecuentemente utilizada a modo de complemento en otras aplicaciones similares como, por ejemplo, en Hootsuite Insights.

2.3. Sprout Social

Sprout Social [9] es otra de las alternativas más populares en cuanto a escucha social para negocios. Ofrece servicios similares a las propuestas anteriores, pero incluyendo todas las funcionalidades en una única plataforma. A parte de la escucha, el paquete ofrece herramientas para programar publicaciones futuras integrando diversas redes sociales y utilizando un algoritmo para optimizar la hora a la que se produzcan estas publicaciones. Además, incluye la posibilidad de interactuar en todas las redes sociales directamente desde la misma aplicación.

En términos de análisis y escucha social, ofrece un paquete de herramientas que permiten utilizar distintas métricas para obtener información relevante de diversas redes sociales (incluyendo Twitter, Facebook, Instagram, Reddit, Tumblr, etc.); pudiendo ver los resultados de cada una de estas al mismo tiempo y comparativamente. En estos análisis, Sprout Social puede identificar automáticamente el comportamiento de las audiencias que se quieran monitorizar y cómo influyen en los temas que son tendencia en cada momento. En lo relativo a la visualización, en lugar de poder observar los resultados de varias métricas en un mismo tablero, Sprout Social cuenta con varias secciones o ventanas que muestran los análisis por separado. En la Figura 2.2, por ejemplo, se puede ver la evolución del número de menciones relativas a distintos temas a lo largo del tiempo y en la Figura 2.3 un análisis del sentimiento general y cómo evoluciona según avanzan los días.

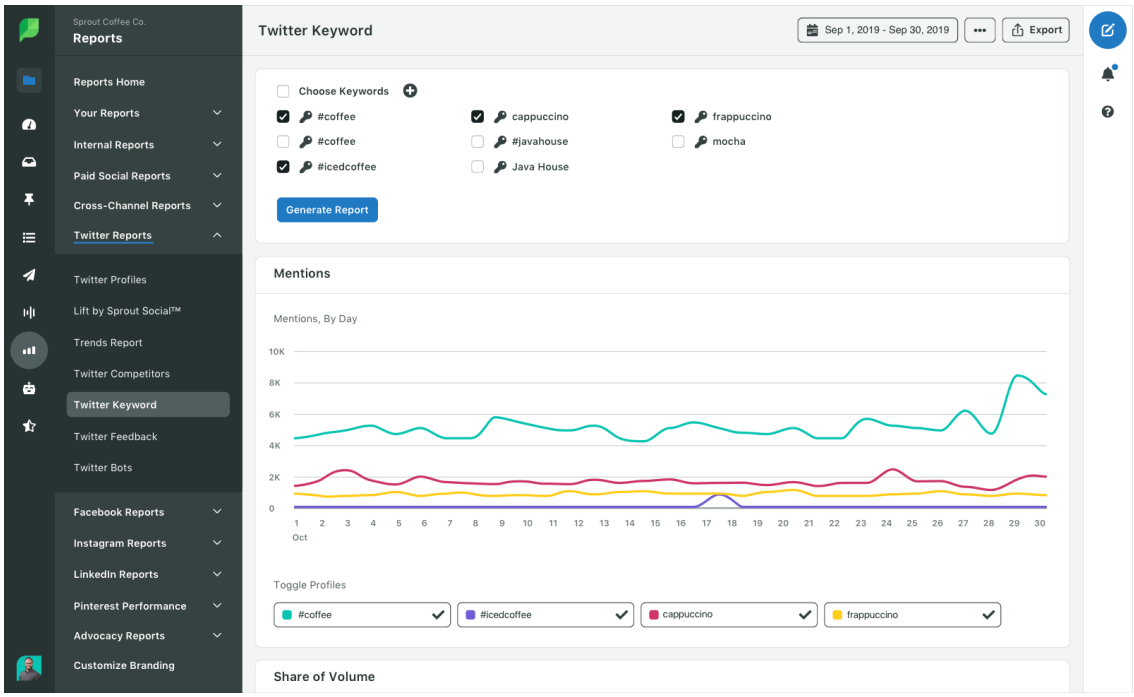

FIGURA 2.2: Análisis de menciones en Sprout Social. Imagen extraída de [10].

2.4. TweetReach

TweetReach [11] es una herramienta de análisis específica para Twitter. Además de contar con la posibilidad de generar informes de alcance, ofrece una versión profesional que permite monitorizar modas, gráficos con datos utilizando distintas métricas, archivos de tuits en los que se puede buscar, la posibilidad de segmentar los datos o la influencia que tienen los usuarios en esta red social, y todo ello en tiempo real.

La principal característica de esta herramienta, y por la que se ha escogido como referencia para este proyecto, es la manera en la que mide el alcance (o *reach*). Normalmente, el alcance es la capacidad o el rango de algo; en redes sociales, el alcance es el tamaño de

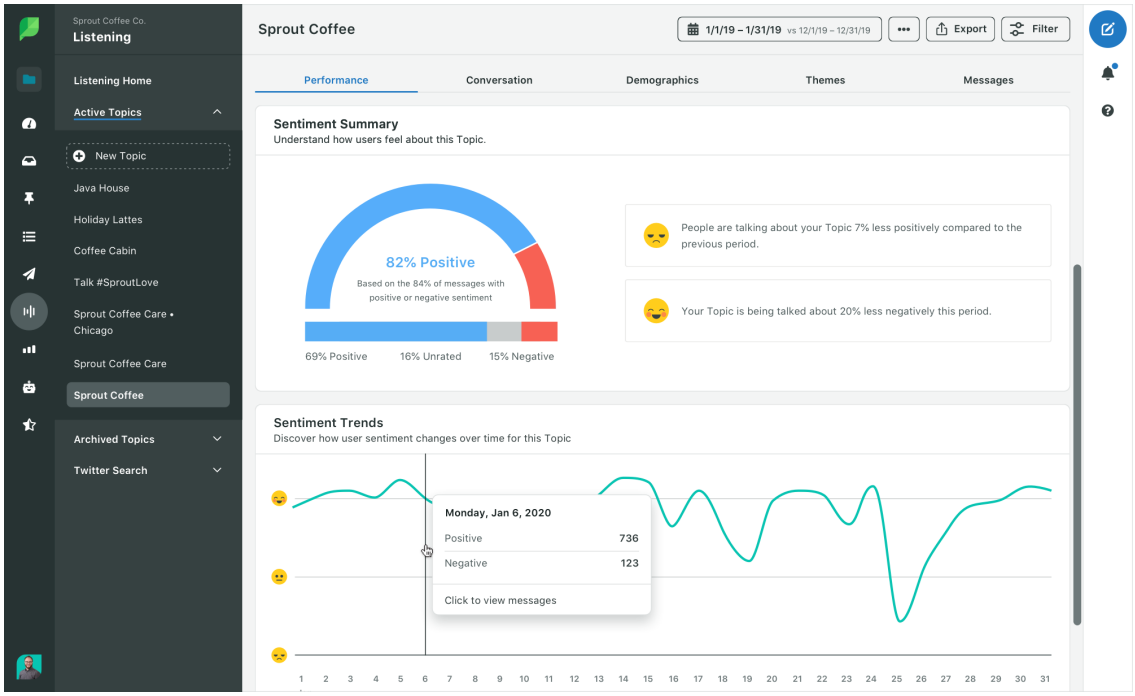

FIGURA 2.3: Análisis de sentimiento en Sprout Social. Imagen extraída de [10].

la potencial audiencia que puede tener un mensaje. En concreto, en un informe de TweetReach, el número que representa el alcance refleja el tamaño de la audiencia de Twitter para la búsqueda específica que haga un usuario. Este número se calcula sumando las cuentas únicas que han recibido el tuit referente el tema buscado, eliminando los identificadores de Twitter duplicados. Por lo tanto, la idea de esta métrica no es aproximar una cifra ni calcular las impresiones totales, sino mostrar el tamaño real de la potencial audiencia.

Como se ha podido comprobar, la mayoría de estas alternativas de escucha social son propietarias y están orientadas al marketing social y a la atención al cliente de empresas. Así, estas pueden hacer un seguimiento de las distintas plataformas que existen y extraer información relevante para conocer mejor a los competidores, tener a sus clientes satisfechos, llegar a más personas y, en definitiva, tomar decisiones empresariales más inteligentes para conseguir que crezca su negocio. Este proyecto pretende tomar este tipo de herramientas como referencia para elaborar una aplicación capaz de hacer análisis similares, y de mostrar el resultado de estos, pero con el objetivo de analizar el impacto que puede llegar a tener una noticia.

Capítulo 3

Análisis

En este capítulo se expondrán las distintas decisiones que se han tomado en la fase de análisis del proyecto. En concreto, los requisitos funcionales y no funcionales, el entorno de trabajo escogido, incluyendo tanto palataformas como herramientas, y las tecnologías necesarias para hacer posible el desarrollo de la aplicación.

3.1. Requisitos

En esta sección se detallarán los requisitos funcionales y no funcionales definidos para este proyecto; describiendo así los servicios que debe proporcionar el sistema, cómo debe reaccionar a una entrada y cómo se debe comportar en situaciones particulares.

3.1.1. Requisitos funcionales

- **RF1.** El usuario deberá poder recopilar un conjunto de tuits que cumplan con los criterios de búsqueda que se especifiquen y almacenar estos localmente.
  - **RF1.1.** El usuario deberá introducir las palabras clave referentes a la noticia que quiera analizar.
  - **RF1.2.** El usuario deberá especificar el nombre del fichero en el que se guardará la colección de tuits en formato JSON.
  - **RF1.3.** El usuario deberá poder determinar cuántos tuits quiere recopilar deteniendo la búsqueda cuando lo desee.
- **RF2.** Para poder conocer el impacto de la noticia, la aplicación deberá analizar la colección de tuits utilizando distintas métricas.
- **RF3.** El usuario deberá poder visualizar los resultados del análisis realizado por la aplicación a través de distintas gráficas.
- **RF4.** La aplicación deberá organizar los resultados del análisis agregando la visualización de todas las métricas en un tablero.

- **RF5.** La aplicación deberá utilizar las siguientes métricas o criterios para mostrar la información:
  - **RF5.1.** Análisis temporal
  - **RF5.2.** Análisis de popularidad
  - **RF5.3.** Análisis geográfico
  - **RF5.4.** Análisis de sentimientos
  - **RF5.5.** Información general o adicional
- **RF6.** El análisis temporal deberá contener, en un gráfico lineal, lo siguiente:
  - **RF6.1.** Interacciones a lo largo del tiempo
  - **RF6.2.** Interacciones acumuladas a lo largo del tiempo
- **RF7.** El análisis de popularidad se hará sobre las siguientes piezas de información:
  - **RF7.1.** Palabras
  - **RF7.2.** *Hashtags*
  - **RF7.3.** Usuarios
  - **RF7.4.** Medios
- **RF8.** El análisis de popularidad deberá mostrar los resultados sobre distintos gráficos que permiten visualizar los elementos y sus frecuencias, como diagramas de barras o nubes de palabras.
- **RF9.** El análisis geográfico deberá mostrar las ubicaciones desde las que se han enviado los tuits en un mapa.
- **RF10.** El análisis de sentimientos se dividirá en dos partes:
  - **RF10.1.** Análisis del sentimiento positivo o negativo del conjunto de tuits.
  - **RF10.2.** Análisis de las emociones presentes en el conjunto de tuits. Para el análisis se deberán tener en cuenta las siguientes emociones:
    - **RF10.2.1.** Enfado
    - **RF10.2.2.** Anticipación
    - **RF10.2.3.** Disgusto
    - **RF10.2.4.** Miedo
    - **RF10.2.5.** Alegría
    - **RF10.2.6.** Tristeza
    - **RF10.2.7.** Sorpresa
    - **RF10.2.8.** Confianza
- **RF11.** El análisis del sentimiento positivo negativo se deberá poder visualizar en un indicador.
- **RF12.** El análisis de las emociones se deberá poder visualizar en un gráfico radial.

- **RF13.** La aplicación deberá mostrar información general sobre los tuits recopilados en una tabla como, por ejemplo:
  - **RF13.1.** Texto completo
  - **RF13.2.** Número de interacciones
  - **RF13.3.** Numbre del autor
  - **RF13.4.** Número de seguidores del autor
  - **RF13.5.** Número usuarios seguidos por el autor
  - **RF13.6.** Fecha del envío
  - **RF13.7.** Cualquier otro dato que se considere relevante
- **RF14.** La aplicación podrá mostrar información adicional a lo largo del desarrollo de la herramienta, siempre que se considere relevante para el análisis del impacto que pueda tener una noticia.

3.1.2. Requisitos no funcionales

- **RNF1.** La aplicación deberá tener un diseño que garantice la adecuada visualización de toda la información de manera intuitiva e interactiva para el usuario final.
- **RNF2.** La aplicación deberá ser escalable para poder incorporar nuevas funcionalidades y mejoras en un futuro.
- **RNF3.** La aplicación deberá ser multiplataforma y visualizarse en distintos navegadores o sistemas operativos.

3.2. Entorno de trabajo

En esta sección se detallarán las diferentes plataformas y herramientas de trabajo que se han decidido utilizar, de acuerdo con los requisitos especificados, para el desarrollo de este proyecto.

3.2.1. Plataformas

Windows

Windows 10 es la versión actual (desde 2015) del sistema operativo desarrollado por Microsoft, es decir, se trata de software propietario. La principal razón por la que ha sido escogido es la comodidad que supone esto último, además de ser el sistema operativo que ya estaba instalado en la máquina en la que se ha desarrollado este proyecto y el más utilizado en todo el mundo, ocupando más de un 75 % de la cuota de mercado de sistemas operativos de escritorio para usuarios [12]. Por último, no ha existido ningún

tipo de problema relacionado con la compatibilidad de las herramientas o las tecnologías escogidas para el desarrollo del proyecto.

PyCharm

PyCharm Community [13] es la versión gratuita del entorno de desarrollo integrado de JetBrains y la plataforma elegida para realizar toda la codificación del proyecto, ya que se ha elegido Python 3 como lenguaje para el desarrollo de este. Las ventajas principales de esta elección son las funcionalidades que ofrece, puesto que facilitan y acortan el proceso de implementación. Algunas de ellas son la asistencia inteligente a la edición de código, las herramientas integradas, como la depuración o el control de versiones, o la facilidad para instalar las tecnologías escogidas para el desarrollo del proyecto.

3.2.2. Herramientas

Git

Git es un sistema de código abierto y libre para el control de versiones de código. El objetivo de este es mantener un registro de todas las modificaciones y avances que se vayan haciendo durante toda la etapa de desarrollo del proyecto, además de permitir compartir y coordinar el trabajo con otras personas. En concreto, se ha utilizado el repositorio de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid, basado en GitLab [14].

Google Forms

Se ha elegido Google Forms para la elaboración del cuestionario de satisfacción, necesario para obtener los resultados que se expondrán en el quinto capítulo sobre la utilidad y la satisfacción de los usuarios con respecto a la aplicación. Se trata de la plataforma de Google para administrar encuestas en línea [15]. Las razón principal por la que se ha escogido es la facilidad para la elaboración de estas encuestas y distribuirlas posteriormente, además de ser gratuita y de poder exportar los resultados y las estadísticas de las respuestas de manera intuitiva.

3.3. Tecnologías

En esta sección se detallarán las tecnologías que han sido escogidas para el desarrollo de este proyecto. Como se ha explicado en la sección anterior, se ha decidido utilizar Python para la implementación de esta aplicación; por lo tanto, la mayoría de estas tecnologías son bibliotecas de código que permiten cumplir con los requisitos definidos previamente.

3.3.1. Recopilación de datos

Tweepy

Tweepy [16] es una biblioteca de código abierto y libre de Python que permite el acceso a la API de Twitter. En vez de tener que codificar las peticiones directamente a la API, este *wrapper* facilita el uso de numerosas funcionalidades que proporciona esta interfaz. Además, de todos los paquetes disponibles que permiten el acceso a la API de Twitter, es el más popular, ya que es la biblioteca más madura y la que cuenta con un mayor número de contribuciones [17] y, por tanto, una comunidad más activa, lo que puede facilitar el desarrollo en caso de que surgieran problemas.

Cuenta con funcionalidades que permiten interactuar en la red social como tuitear, seguir o dejar de seguir a otros usuarios, borrar tuits o descargar los mensajes publicados en el *timeline* de cualquier usuario. Sin embargo, la funcionalidad que se necesita para conseguir la recopilación de un conjunto de tuits, que es uno de los objetivos principales de este proyecto, es la de búsqueda, que puede hacerse sobre mensajes ya enviados en Twitter o a través de un flujo en tiempo real que comienza en el momento que se lanza la búsqueda. Ambos casos permiten introducir una consulta con palabras clave, además de otros parámetros, como se deseaba en un comienzo para esta aplicación.

jsonpickle

jsonpickle [18] es una biblioteca de Python especializada en la serialización de objetos a formato JSON y en el proceso inverso de decodificación. En este proyecto esos objetos serán los tuits recopilados. La principal razón por la que se ha escogido esta tecnología es que las librerías estándares de Python para codificar en JSON (como json, simplejson o demjson) solo pueden manejar tipos primitivos de Python con equivalente JSON directo como, por ejemplo, listas o diccionarios. jsonpickle, por el contrario, permite serializar estructuras de datos más complejas.

3.3.2. Análisis y visualización

pandas

pandas [19] es una biblioteca de código abierto y libre de Python desarrollada como extensión de NumPy y especializada en el análisis y el tratamiento de datos. La principal ventaja por la que se ha escogido esta tecnología es la facilidad que ofrece para manipular datos a través de *dataframes*, además de contar con herramientas para leer y escribir estos datos en distintos tipos de archivos. En este proyecto, se leerán los tuits almacenados en un archivo JSON y se irá preparando y segmentando la información en distintas tablas, dependiendo de la métrica, para poder hacer los análisis pertinentes.

NumPy

NumPy [20] es una biblioteca de código abierto y libre de Python especializada en computación numérica creada en 2005. Esta tecnología ofrece numerosas funcionalidades y se caracteriza por ser rápida y versátil, cuando se indexan vectores de N dimensiones, por ejemplo, y por ser rápida y eficiente al contar con un núcleo escrito y optimizado en C, además de ser, a la vez, sencilla de utilizar gracias a su sintaxis de alto nivel. Se ha escogido para facilitar el tratamiento de información extraída de los tuits, almacenados previamente en *dataframe*, para realizar cálculos o poder generar representaciones gráficas.

Matplotlib

Matplotlib [21] es una biblioteca de Python especializada en la construcción de gráficas que permitan visualizar los datos contenidos en un array de NumPy. Existen numerosos parámetros para configurar las figuras: títulos, formas, colores, tamaños, valores que se quieran introducir, tipo de gráfico, etc. Por lo tanto, las posibilidades de generación de estas figuras son ilimitadas y, por esta misma razón, esta tecnología puede resultar de utilidad para el objetivo de visualizar los resultados de los análisis. Sin embargo, las gráficas generadas se exportan como imágenes, no como un gráfico dinámico e interactivo, que es también uno de los objetivos que se buscan en este proyecto.

Plotly

Plotly [22] es una biblioteca de código abierto y libre de Python que permite trazar figuras que, a diferencia de Matplotlib, pueden ser interactivas. Soporta más de 40 tipos de gráficos que cubren un amplio espectro de casos de uso estadísticos, financieros, geográficos científicos y tridimensionales. Es por esta razón, y por la posibilidad de que estos sean dinámicos, por la que se ha seleccionado esta tecnología, junto con Dash, para la elaboración de la mayor parte del tablero que contendrá las representaciones de los resultados del análisis de la colección de tuits.

Dash

Dash [23] es un *framework* de Python que permite construir aplicaciones web. Está escrito sobre Flask, Plotly.js y React.js, es decir, Dash, a través de unos sencillos patrones, permite abstraer todas estas tecnologías y protocolos para crear una aplicación web dinámica, lo que convierte a Dash en la elección ideal para la construcción de herramientas que permitan visualizar datos con interfaces altamente configurables utilizando exclusivamente Python. Además, estas aplicaciones son renderizadas en el navegador, por lo que son inherentemente multiplataforma, y pueden ser desplegadas tanto localmente como en un servidor; si se quisieran compartir con otros usuarios a través de una URL. Otra razón por la esta tecnología ha sido escogida es la posibilidad de ser utilizada en conjunción con Plotly para generar los gráficos que se mostrarán en la aplicación.

WordCloud

WordCloud [24] es una biblioteca de Python desarrollada por Andreas C. Müller, investigador asociado en la Universidad de Columbia, especializada en crear nubes de palabras para la representación gráfica de texto. Permite visualizar una lista de palabras cuya relevancia se muestra con un color y un tamaño diferente, como se puede visualizar en la Figura 3.1. En este proyecto, esta relevancia se medirá en función de la frecuencia con la que aparezca el elemento que se está analizando en la colección de tuits. Comparada con otras bibliotecas, esta tiene claras ventajas que el mismo autor destaca en la documentación: rellena todo el espacio posible, permite utilizar distintas máscaras, cuenta con un simple y eficiente algoritmo que puede ser modificado fácilmente y está desarrollada en Python. Finalmente, para poder generar estas figuras es necesario el uso de Matplotlib.

FIGURA 3.1: Ejemplo de nube de palabras

NLTK

NLTK [25] (Natural Language Toolkit) es una biblioteca de Python que facilita un conjunto de herramientas para procesamiento de lenguaje natural. Proporciona interfaces fáciles de utilizar para acceder a más de 50 corpus y recursos léxicos, además de otras bibliotecas que permiten procesar texto clasificar, lematizar o etiquetar texto, entre otras funcionalidades. En concreto, se ha escogido esta tecnología para acceder a su corpus de *stop words* o palabras vacías, ya que serán necesarias para acelerar el procesamiento y evitar palabras sin significado cuando se analicen los textos de los tuits y se determine la carga emocional o se mida la frecuencia de elementos como palabras o *hastags*.

EmoLex

EmoLex es una lista de palabras y sus asociaciones con ocho emociones diferentes (enfado, miedo, anticipación, confianza, sorpresa, tristeza, alegría y disgusto) y dos sentimientos (positivo y negativo). Este lexicón fue desarrollado por el National Research Council de Canadá [26] porque, a pesar el análisis de la polaridad de las palabras habría cobrado importancia, los lexicones disponibles hasta el momento eran limitados y contaban con menos emociones. Esta lista de palabras se desarrolló originalmente en inglés,

pero en 2017 se tradujo a más de cien lenguas utilizando el servicio de traducción de Google.

googletrans

googletrans [27] es una biblioteca de Python que implementa la API de Google Translate de manera gratuita. La principal razón por la que se ha escogido esta tecnología es la falta de precisión en la traducción al castellano de EmoLex. Tras analizar su uso se determinó que el acierto es mayor cuando se traduce el texto de los tuits de la colección al inglés, en caso de que no sea este el idioma empleado, y se utiliza el lexicón original. Además, esta biblioteca cuenta con otras ventajas como la velocidad, la fiabilidad o la detección automática de idioma.

GeoPy

GeoPy [28] es una biblioteca de Python que funciona como cliente para numerosos servicios web de geocodificación populares. Facilita el proceso de asignar coordenadas geográficas a cadenas de caracteres que contengan direcciones, ciudades, países u otros puntos de referencia en todo el mundo; con el objetivo de situarlas posteriormente en un mapa. Sabiendo que el porcentaje de tuits que contienen las coordenadas desde las que se han enviado es muy bajo, esta tecnología resulta de gran utilidad para obtener la ubicación de los usuarios que tuitean estos mensajes, ya que este dato está presente con más frecuencia.

Capítulo 4

Diseño y desarrollo

En este capítulo se expondrán las distintas decisiones que se han tomado en la fases de diseño y desarrollo del proyecto. En concreto, sobre la arquitectura de aplicación, la recopilación de datos y el análisis y la visualización de estos.

4.1. Arquitectura de la aplicación

En la Figura 4.1 podrá observarse la arquitectura general de la aplicación desarrollada en este proyecto. Esta se despliega en un servidor y cuenta con dos módulos: *twitter_client.py*, que permite al usuario introducir una consulta para extraer la información de Twitter y guardarla en un archivo en formato JSON, y *app.py*, que realiza el análisis de la colección de tuits extraída y permite al usuario visualizar los resultados de este análisis a través de un navegador web.

4.2. Recopilación de datos

En esta sección se detallarán las distintas decisiones de diseño que se han ido tomando a lo largo del proyecto y cómo se ha desarrollado la parte referente a la recopilación de tuits para su futuro análisis.

4.2.1. Twitter API: Autentificación

La API de Twitter [29] maneja una cantidad enorme de datos. Es por eso que la manera de asegurarse de que estos datos estén seguros y puedan ser utilizados por desarrolladores es a través de la autentificación. Existen diversos métodos para ello, los más populares son OAuth 1.0a y OAuth 2.0 Bearer Token. Algunas de las diferencias entre las funcionalidades que ofrecen estos estos dos métodos se pueden observar en la Tabla 4.1.

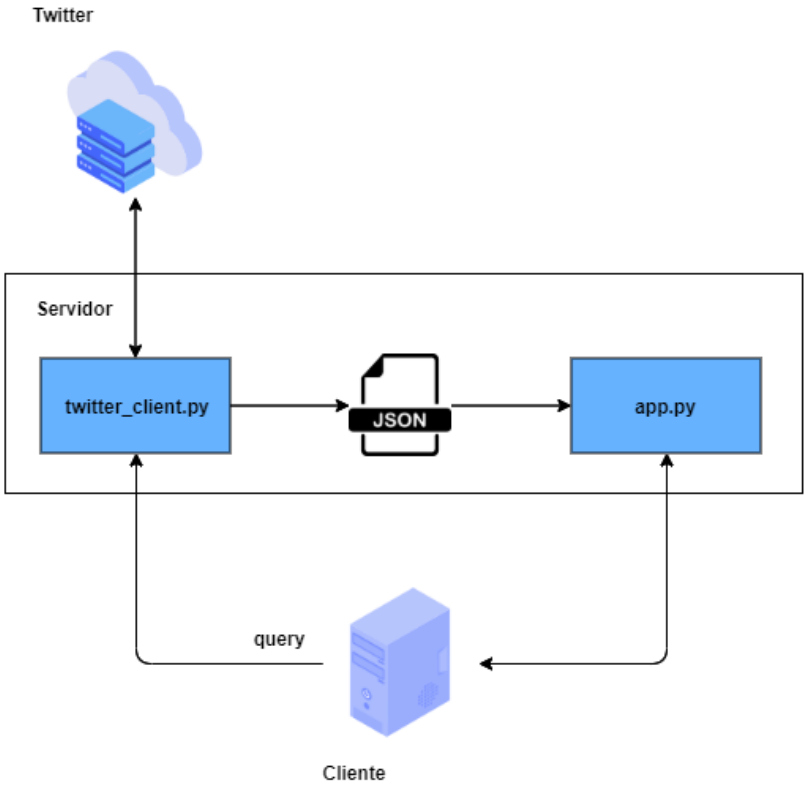

FIGURA 4.1: Arquitectura de la aplicación.

TABLA 4.1: Diferencias entre OAuth 1.0a y OAuth 2.0 Bearer Token

Caso de uso	**OAuth 1.0a**	**OAuth 2.0**
Búsqueda de tuits	Sí	Sí
Obtener *timeline* de un usuario	Sí	Sí
Obtener datos de tendencias	Sí	Sí
Publicar, marcar como «Favorito» o retuitear un tuit	Sí	No
Obtener la dirección de correo electrónico de un usuario	Sí	No
Leer o escribir datos de anunciantes	Sí	No

En resumen, la diferencia principal entre estos dos métodos es que, siempre que la aplicación desarrollada esté autorizada por Twitter, OAuth 1.0a permite tener acceso a la información privada de una cuenta y realizar acciones de Twitter con esa cuenta. Sin embargo, OAuth 2.0 Bearer Token permite únicamente tener acceso a información pública disponible en Twitter.

OAuth 1.0a

Muchas de las aplicaciones implementadas bajo la plataforma de desarrollo de Twitter utilizan OAuth 1.0a por el interés que supone poder interactuar en la red social con una cuenta, además de poder recopilar información. Sin embargo, para llevar a cabo esto es necesario que el usuario de esa cuenta autentifique la aplicación firmando cada petición que se realice la API de Twitter incluyendo las siguientes claves y *tokens*:

- ***Consumer API key***
- ***Consumer API secret key***
- ***Access token***
- ***Access token secret***

Las dos primeras representan la aplicación desarrollada cada vez que se hagan peticiones a la API. Sin embargo, los *tokens* de acceso son específicos para un usuario, es decir, son solo necesarios cuando se decida autentificarse utilizando el método OAuth 1.0a, ya que representan la cuenta a través de la que se hace la petición.

OAuth 2.0 Bearer Token

OAuth 2.0 Bearer Token es el método de autentificación específico para aplicaciones en las que no estén involucrados usuarios, es decir, para desarrolladores que necesitan solo acceso para leer información pública, como se ha explicado anteriormente. Por lo tanto, no serán necesarios los *tokens* de acceso que necesitaba OAuth 1.0a para autentificar una cuenta. Sin embargo, sí será necesario un *bearer token*. Para conseguir este *bearer token* serán necesarias las claves de consumidor que identifican la aplicación. De esta manera, el proceso de autentificación a través de este método sigue los siguientes pasos:

1. La aplicación codifica su clave y secreto de consumidor en un conjunto de credenciales especialmente codificadas.
2. La aplicación hace una petición al *endpoint* correspondiente para intercambiar estas credenciales por el *bearer token*.
3. Cuando se acceda a la API REST, la aplicación utiliza el *bearer token* para autentificarse.

Al no existir la necesidad de de firmar la petición, este método es mucho más simple que el modelo estándar (OAuth 1.0a).

Límite de velocidad

Otro de los principales factores a tener en cuenta cuando se trabaja con la API de Twitter es el límite de velocidad a la que se pueden hacer peticiones. Esta tasa depende del *endpoint* que se quiera utilizar y siempre se divide en intervalos o ventanas de 15 minutos, independientemente del valor del límite. Además, las peticiones que utilizan *tokens* de acceso, es decir, las hechas en nombre de usuarios, no agotan el límite que existe para las peticiones que se envían utilizando el método OAuth 2.0 Bearer Token para autentificarse y viceversa.

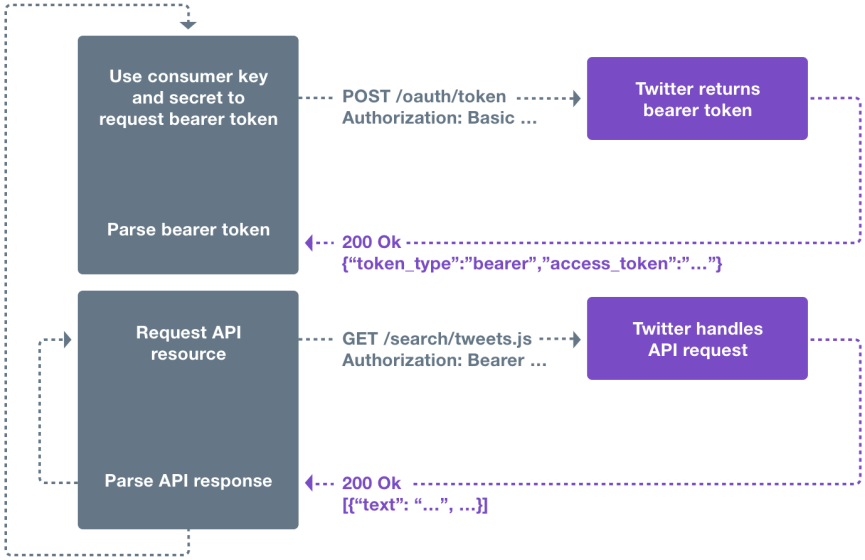

FIGURA 4.2: Flujo de autentificación utilizando OAuth 2.0 Bearer Token. Imagen extraída de [30].

4.2.2. Tweepy: Búsqueda

Tweepy, como se ha detallado en la sección Recopilación de datos del capítulo anterior, es biblioteca de código de Python que encapsula y facilita la comunicación con la API de Twitter que ha sido escogida para el desarrollo de este proyecto.

Existen dos maneras de afrontar la recopilación de datos: a través de una búsqueda sobre tuits ya publicados en la red social [31] o filtrando tuits enviados en tiempo real [32]; ambos métodos tienen sus ventajas y limitaciones.

Buscar tuits

La API de Twitter ofrece tres niveles para la búsqueda de tuits: *Standard*, *Premium* y *Enterprise*. A pesar de las limitaciones, la opción estándar es la única viable para este trabajo, ya que es la única pública y gratuita. Este *endpoint* solo permite obtener tuits publicados en los últimos siete días y la fidelidad de datos no es del todo completa. Las otras opciones, por el contrario, permiten recopilar mensajes enviados desde 2006 y con una fidelidad de datos completa. Además, el nivel estándar funciona de manera similar, pero no igual, a la búsqueda disponible en el cliente web o móvil de Twitter. Por lo tanto, permite recibir peticiones que contengan búsquedas sencillas de tuits recientes o populares. Es importante destacar que la versión estándar se centra en la relevancia y no en la completitud. Esto quiere decir que ciertos tuits o usuarios pueden faltar en los resultados de la búsqueda.

Filtrar tuits en tiempo real

En este caso, la API de Twitter ofrece dos niveles: *Standard* y *Enterprise*. Las principales diferencias entre ambos es la posibilidades de filtrado y la gestión de las conexiones. La versión estándar solo permite un filtro en una conexión habilitada, siendo necesaria una desconexión si se quisiera cambiar el filtro. El segundo nivel, por el contrario, permite miles de filtros en una sola conexión y no es necesaria una desconexión para añadir o quitar filtros. Sin embargo, para el objetivo de este proyecto, la versión estándar es suficiente.

Tweepy facilita clases y métodos para implementar ambas opciones y, por ello, se han desarrollado ambas en este trabajo. Sin embargo, a pesar de que los datos obtenidos a través de un flujo filtrado de tuits en tiempo real son más completos, se ha escogido la opción de buscar sobre los ya publicados en los últimos siete días. A pesar de esta limitación, se ha tomado esta decisión porque, para desarrollar el análisis y la visualización de una colección de tuits, la obtención de datos a través de este método es más flexible y requiere menos tiempo.

Optimización de la búsqueda

Aunque la recopilación de datos a través de la búsqueda estándar de Twitter no sea completa, existen estrategias para maximizar los recursos que nos puede ofrecer la API. A continuación, se expondrán las decisiones de diseño que se han tomado en este trabajo para poder encontrar toda la información posible en el menor tiempo posible manteniéndose, aun así, dentro de los límites de la API:

1. Aumentar la tasa de peticiones

Inicialmente, el límite de la velocidad de la API expuesto en la documentación inicial para aplicaciones que utilicen autentificación de usuario (a través de OAuth 1.0a), que es el método más popular, es de 180 peticiones en cada ventana de 15 minutos. Además, por cada petición se pueden solicitar un máximo de 100 tuits, lo que supone una tasa total de 18000 tuits cada 15 minutos.

La solución a este problema es utilizar la autentificación específica para aplicaciones (a través de OAuth 2.0 Bearer Token). Utilizar este método implica disponer de un límite de velocidad mayor, aumentando la tasa a 450 peticiones en cada ventana de 15 minutos. Con una limitación de 100 tuits por petición, la tasa total supone 45000 tuits cada 15 minutos, valor 2,5 veces mayor que la opción que se planteó inicialmente. Además, dados los requisitos concretos de este proyecto, la funcionalidad adicional que ofrece la API de Twitter autentificándose con *tokens* de acceso no es necesaria, ya que solo se requiere acceder a información pública disponible en esta red social.

En Tweepy, esta decisión se traduce en utilizar *AppAuthHandler* en lugar de la opción más popular, *OAuthHandler*:

```
def authenticate():
    return AppAuthHandler(credentials_twitter.API_KEY,
        credentials_twitter.API_SECRET_KEY)

def get_client():
    auth = authenticate()
    cliente = API(auth, wait_on_rate_limit=True,
        wait_on_rate_limit_notify=True)

    if not cliente:
        print("No se pudo autenticar")
        sys.exit(-1)

    return cliente
```

API_KEY y API_SECRET_KEY son las claves del consumidor. Como se puede observar, no son necesarios los *tokens* de acceso para autentificarse. Además, como opciones adicionales se han activado los *flags wait_on_rate_limit* y *wait_on_rate_limit_notify*. Esto permite a la aplicación esperar automáticamente cuando alcance el límite de peticiones y continúe cuando comience la siguiente ventana. De este modo, se evita tener que programar esta funcionalidad manualmente.

2. Mantener el contexto de la búsqueda entre ventanas de tiempo

Las API REST son *stateless*, es decir, no existe un contexto implícito que se mantenga por el servidor entre llamadas consecutivas a la misma API, de manera que esta pueda saber qué resultados se han enviado ya al cliente. Por lo tanto, la solución a este problema es decir directamente a la API dónde se encuentra el contexto de la búsqueda, a partir de resultados previos, para que el servidor pueda entonces enviar el siguiente conjunto de resultados; este proceso se denomina paginación. La API de Twitter para buscar tuits permite esto a través de dos parámetros de entrada: *since_id* y *max_id*; que sirven como los límites inferior y superior de los identificadores únicos que Twitter asigna a los mensajes, respectivamente. Así, variando estos dos valores durante sucesivas llamadas a la API en un bucle se pueden paginar los resultados.

3. Ampliar el tamaño la colección

Finalmente, la última cuestión consiste en afrontar el problema de la falta de completitud en la búsqueda estándar de Twitter. No todos los tuits que cumplan con los criterios establecidos de búsqueda. Sin embargo, de la gran cantidad de tuits que se pueden obtener, muchos serán retuits. Existe la posibilidad que los tuits originales, contenidos en estos retuits, no estén presentes en el conjunto recopilado inicialmente. Además, cada retuit codifica en su contenido el tuit original en formato JSON. Por lo tanto, si extraemos estos tuits originales y los añadimos a la colección, podemos ampliar el tamaño de la

colección original. Este proceso es sencillo de implementar, ya que cada tuit tiene asignado un identificador único, lo que permite incluir solo aquellos que falten en el conjunto inicial y no añadir duplicados. Esta mejora aumenta el número de tuits en entre un 2 % y un 10 %, aumentando la completitud en cierta medida y haciendo que la colección solo contenga tuits originales y no retuits.

Criterios de búsqueda

Para finalizar esta sección, se van a concretar los parámetros se han utilizado al realizar una búsqueda a través de la API de Twitter:

- ***q:*** parámetro obligatorio que contiene la consulta introducida por el usuario de la aplicación. La consulta es una cadena con una longitud máxima de 500 caracteres y codificación UTF-8. Los resultados de la búsqueda están condicionados por lo precisa que pueda llegar a ser esta, es decir, cuanto más vaga sea la consulta, existirán más posibilidades de que entre ruido en el conjunto de tuits que se recopilen. Para acotar los resultados el usuario tendrá que utilizar los operadores de búsqueda estándar de Twitter [33].
- ***count:*** parámetro opcional que permite definir el número de tuits que se quieren recibir por cada petición. El máximo es 100, que es el valor utilizado en la aplicación, ya que el valor por defecto es 15.
- ***result_type:*** parámetro opcional que permite escoger qué tipo de resultados se quieren obtener con la búsqueda; existen tres valores:
  - ***mixed:*** los resultados incluyen tanto tuits populares como tuits recientes.
  - ***recent:*** los resultados solo incluyen los tuits recientes.
  - ***popular:*** los resultados solo incluyen los tuits populares.
- ***max_id:*** parámetro opcional que permite obtener tuits cuyo identificador es menor (es decir, más antiguos) o igual al valor especificado. Este parámetro se va actualizando en cada ventana como se ha explicado anteriormente.
- ***since_id:*** parámetro opcional que permite obtener tuits cuyo identificador es mayor (es decir, más recientes) o igual al valor especificado. Este parámetro se va actualizando en cada ventana como se ha explicado anteriormente. Además, como existe un límite en el número de tuits que se pueden obtener con la API, si este parámetros supera ese límite, pasará a valer automáticamente el identificador más antiguo que esté disponible.
- ***tweet_mode:*** parámetro opcional añadido posteriormente; cuando Twitter permitió que los mensajes pudieran contener más de 140 caracteres. Asignando el valor *extended*, los tuits devueltos por la API contendrán el texto original completo. Por

defecto, si no se especifica esta opción, los mensajes contienen siempre 140 caracteres, es decir, los que tienen una longitud mayor son acortados. Por lo tanto, ciertas métricas del análisis que se quiere hacer en este proyecto serían menos precisas.

A pesar de que existen más parámetros, como la geolocalización o el idioma, que permiten acotar aun más los resultados, no han sido necesarios para la aplicación que se ha desarrollado en este proyecto.

En cuanto al parámetro *result_type*, la decisión original era utilizar la opción por defecto (*mixed*) y obtener una colección suficientemente grande de tuits para poder hacer un análisis exhaustivo. Sin embargo, durante el desarrollo de la parte de análisis y visualización, descrito en la siguiente sección, la máquina en la que se ha desarrollado este trabajo no soportaba procesar los resultados de las primeras búsquedas, que contenían más de 150000 mensajes. Además, el hecho de que contuvieran tuits recientes hacía que, aunque se usara una pequeña porción de la colección, los tuits más antiguos de ese subconjunto eran del día anterior al día en el que se realizó la búsqueda, lo que dificultaba realizar el análisis de las interacciones a lo largo del tiempo, por ejemplo. Por lo tanto, y en definitiva, para el desarrollo de la parte restante del proyecto se ha utilizado la opción *popular* en el parámetro *result_type*, ya que al recopilar solo los tuits más populares, sí que se obtenían mensajes de hasta 7 días de antigüedad y los tamaños de estas colecciones no superaban los 150 tuits, perdiendo precisión, pero facilitando el desarrollo del proyecto.

4.3. Análisis y visualización

En esta sección se detallarán las distintas decisiones de diseño que se han ido tomando a lo largo del proyecto y cómo se ha desarrollado la parte referente al análisis de los tuits recopilados, utilizando diversas métricas, y a la visualización de los resultados de este con distintas gráficas.

4.3.1. Tweet Object

Cada tuit devuelto por la API de Twitter está codificado en formato JSON [34] (*JavaScript Object Notation*). Este formato se caracteriza por almacenar la información relativa a un objeto en parejas de claves y valores, donde las claves son atributos ya definidos que se utilizan para definir el objeto. Además, Twitter cuenta con diversos tipos de objetos. El objeto tuit, por ejemplo, ouede contener un autor, un mensaje, un identificador único, la fecha en la que se publicó o la coordenadas geográficas; mientras que el objeto usuario contiene un nombre, un identificador único, el número de seguidores, una descripción, etc. A veces, los tuits pueden contener también, dentro del atributo *entities*, que es a su vez otro objeto, los *hashtags*, las menciones o los enlaces que contiene el mensaje. En total, un objeto tuit puede encapsular más de 150 atributos asociados. En este trabajo se han

decidido utilizar únicamente los más relevantes y necesarios para las métricas definidas en la fase de análisis del proyecto. En el Anexo A se puede ver un ejemplo del *payload* en JSON de un tuit.

4.3.2. Métricas

A continuación, se detallará cómo se han desarrollado cada una de las métricas, que constituyen el análisis de un conjunto de tuits, y sus respectivas representaciones gráficas.

Tratamiento de datos

Antes de realizar cada una de las métricas tiene lugar un tratamiento previo de los datos a través de la función *generate_dataframe*, que, a partir de una lista de tuits extraídos del fichero JSON donde se han almacenado tras la búsqueda, construye un *dataframe* utilizando la biblioteca de Python pandas. Solo se seleccionan los atributos que son necesarios para las métricas, como se puede ver en el siguiente fragmento de código:

```
1  def generate_dataframe(data):
2      # Creamos dataframe con atributos necesarios para el análisis
3      df = pd.DataFrame(data=[tweet["created_at"] for tweet in data
           ], columns=["Fecha"])
4      df["ID"] = np.array([tweet["id"] for tweet in data])
5      df["Usuario"] = np.array([tweet["user"]["screen_name"] for
           tweet in data])
6      df["Seguidores"] = np.array([tweet["user"]["followers_count"]
            for tweet in data])
7      df["Seguidos"] = np.array([tweet["user"]["friends_count"] for
            tweet in data])
8      df["Verificado"] = np.array([tweet["user"]["verified"] for
           tweet in data])
9      df["Tweet"] = np.array([borrar_urls(tweet["full_text"]) for
           tweet in data])
10     df["Retweets"] = np.array([tweet["retweet_count"] for tweet
           in data])
11     df["Favoritos"] = np.array([tweet["favorite_count"] for tweet
            in data])
12     df["Fuente"] = np.array([tweet["source"] for tweet in data])
13     df["Ubicación"] = np.array([tweet["user"]["location"] for
           tweet in data])
14     df["Hashtags"] = np.array([tweet["entities"]["hashtags"] for
           tweet in data])
15     df["Menciones"] = np.array([tweet["entities"]["user_mentions
           "] for tweet in data])
16     df["URLs"] = np.array([tweet["entities"]["urls"] for tweet in
            data])
17
18     # Ajustes en el dataframe
19     df["Fecha"] = pd.to_datetime(df["Fecha"])
20     df["Fecha"] = df["Fecha"].dt.strftime("%Y-%m-%d %H:%M:%S")
21     df.loc[df["Fuente"].str.contains("iPhone"), "Fuente"] = "
           iPhone"
```

```
22      df.loc[df["Fuente"].str.contains("Web"), "Fuente"] = "Web App
            "
23      df.loc[df["Fuente"].str.contains("Android"), "Fuente"] = "
            Android"
24      df.loc[~df["Fuente"].str.contains("Android|iPhone|Web"), "
            Fuente"] = "Otra"
25      df.drop_duplicates(subset="ID", inplace=True)
26
27      return df
```

Si en un futuro se quisieran añadir métricas, solo habría que crear una columna al construir el *dataframe*, es decir, añadir una línea de código más. Finalmente, se han hecho unos ajustes adicionales para depurar el contenido general de los datos:

1. Se ha creado una función auxiliar (*borrar_urls*) que, utilizando la biblioteca de Python *re*, elimina las URLs del atributo que contiene el texto de los mensajes con una expresión regular.
2. Convertir el atributo *created_at*, es decir, la columna «Fecha», que contiene una cadena con el formato de Twitter, a un objeto de tipo *datetime* que pueda entender Python cuando haya que ordenar o representar los datos.
3. Borrar la cadena «Twitter for» del atributo *source*, que contiene la fuente desde la que se ha publicado el mensaje en la red social.
4. Finalmente, se eliminan los tuits que, por cualquier razón, puedan estar duplicados dentro de la colección.

Cualquier otro ajuste particular que se requiera hacer para una métrica en concreto, se hará creando un *dataframe* nuevo, de manera que se puedan tratar siempre los datos sin perder el conjunto original, por si se quisieran hacer cambios en un futuro y mantener un diseño más modular.

Interacciones

Esta métrica muestra cómo evoluciona tanto el número de retuits y como el número de veces que un tuit ha sido marcado como «Favorito» a lo largo del tiempo. El objetivo es que, de este modo, el usuario, que ha buscado previamente mensajes relativos a una noticia concreta, pueda tener una visión general de las veces que se ha interactuado y cuándo se ha hecho. El único tratamiento previo que ha requerido esta métrica ha sido ordenar los tuits de la colección por la fecha de publicación, de manera que pudieran ser representados cronológicamente.

La representación se ha realizado trazando las dos líneas (una por cada tipo de interacción) en un gráfico cuyo eje de abscisas indica la fecha y el eje de ordenadas el número entero de interacciones, como se puede observar en la Figura 4.3. La figura se

ha construido con dos objetos de tipo *Scatter* (de la biblioteca *graph_objects* de Plotly) y se ha integrado como un componente más en el tablero general, elaborado con Dash. Esto permite que el gráfico sea dinámico y se pueda alejar o acercar, descargar una imagen o visualizar el contenido del tuit si se pasa el cursor por encima de un punto.

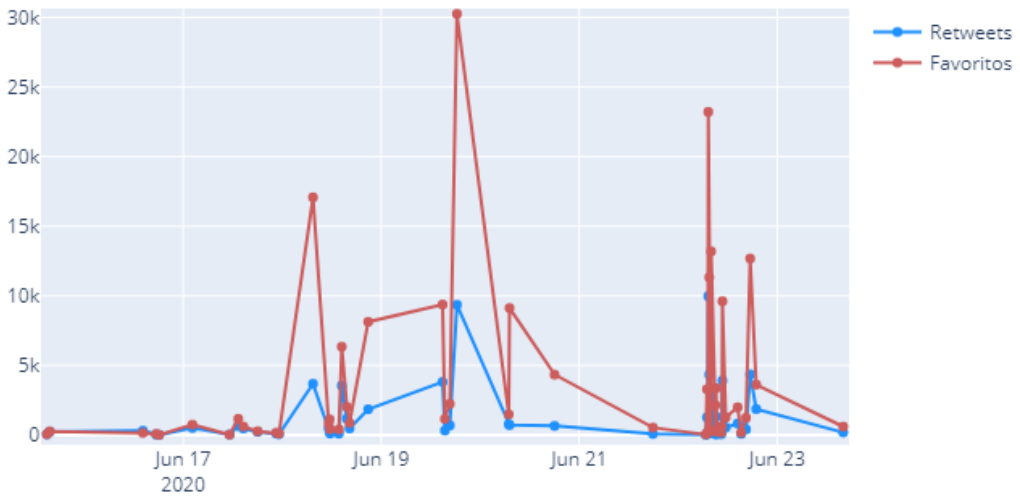

FIGURA 4.3: Visualización de interacciones a lo largo del tiempo

Interacciones acumuladas

Esta métrica, igual que la anterior, muestra también tanto el número de retuits y como el número de veces que un tuit ha sido marcado como «Favorito» a lo largo del tiempo. Sin embargo, en lugar de visualizar la variación en el número de interacciones de cada tuit por se parado, el usuario podrá visualizar la cantidad acumulada a lo largo del tiempo. De este modo, se pueden ver los incrementos y las «mesetas» de manera más intuitiva e intuir en qué fechas ha tenido más impacto la noticia y cuánto.

Para conseguir este resultado se ha seguido un formato similar al de la gráfica anterior, pero alterando ciertos parámetros, como la opción *fill* de Plotly, al construir la figura para visualizar mejor la cantidad total de ambas interacciones. Además, para conseguir obtener la cifra acumulada, se ha utilizado la función *cumsum* de NumPy, que devuelve la suma acumulada a lo largo de un eje dado un array de datos (las columnas que contienen los dos tipos de interacciones dentro del *dataframe*).

Palabras frecuentes

Esta métrica permite visualizar las palabras que aparecen con más frecuencia dentro del texto de todos los mensajes enviados excluyendo, obviamente, *hashtags* y menciones.

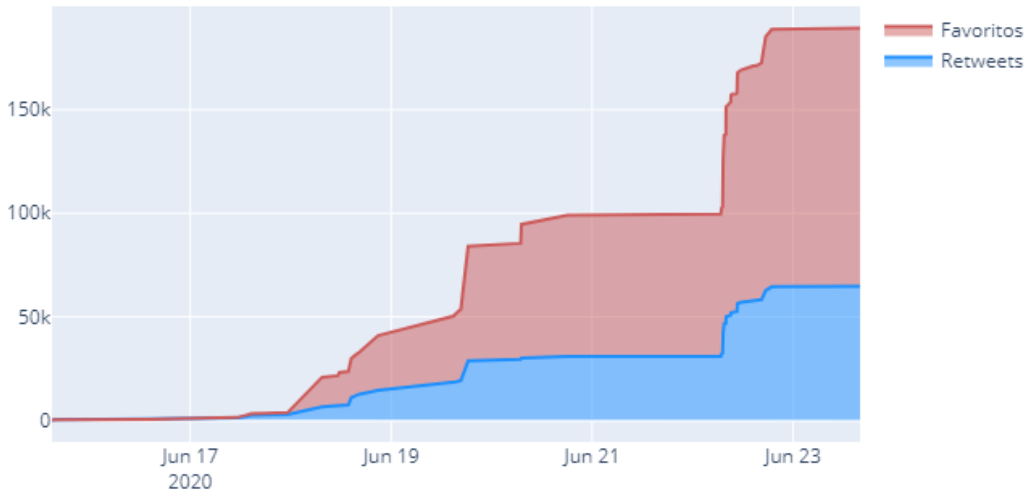

FIGURA 4.4: Visualización de interacciones acumuladas a lo largo del tiempo

Para ello, se ha desarrollado una función auxiliar denominada *generate_dataframe_palabras* que creará, a partir del *dataframe* original, uno nuevo que contará con dos columnas: una con las palabras ordenadas de mayor a menor frecuencia y otra que contendrá la frecuencia correspondiente a cada una de ellas. Finalmente, devolverá las 10 primeras palabras de ese conjunto de datos, es decir, las 10 más frecuentes.

Antes de la construcción de este *dataframe* se ha hecho un tratamiento previo de los datos. Primero, utilizando las funciones *split* y *lower* propias de Python, se separan todas las palabras del conjunto de tuits y se pasan a minúsculas. A continuación, se crea una lista única de términos con la ayuda de la biblioteca de Python *itertools*. A partir de esta lista, se crea el diccionario que servirá para construir el *dataframe* al final utilizando un objeto de tipo *Counter*. Así, este diccionario contendrá todas las palabras como claves y sus frecuencias como valores. Finalmente, se eliminan las palabras vacías o *stop words* y sus respectivas frecuencias de este diccionario iterando sobre el corpus proporcionado por NLTK. Una vez finalizado este proceso y generado el *dataframe*, se genera una nube de palabras utilizando las bibliotecas WordCloud y Matplotlib como la que se puede ver en la Figura 3.1 del capítulo anterior.

Hashtags, meciones y URLs frecuentes

La implementación de estas métricas es, salvo algunos matices que se explicarán más adelante, muy similar. Estas tres piezas de información se pueden encontrar en el atributo *entities* del objeto tuit original, que es también un objeto y tiene la siguiente estructura:

```
"entities": {
  "hashtags": [
        {
      "indices": [26, 32],
      "text": "LGTBI"
    }
  ],
  "urls": [
        {
      "indices": [117, 140],
      "url": "https://t.co/CkAnYa3deY",
      "display_url": "twitter.com/i/web/status/1\u2026",
      "expanded_url": "https://twitter.com/i/web/status
         /1274234708977745925"
    }
  ],
  "user_mentions": [
        {
      "name": "Correos",
      "indices": [36, 44],
      "screen_name": "Correos",
      "id": 198535580,
      "id_str": "198535580"
    }
  ]
}
```

Puede contener más entidades como archivos multimedia o ecuestas, pero se ha decidido tomar estos tres para el análisis. Tanto los *hashtags* como los enlaces y las menciones son también objetos y cuentan con sus propios atributos, que contienen información adicional como, por ejemplo, los índices que marcan la posición de estos elementos dentro del tuit. En concreto, se han utilizado los atributos *text*, *expanded_url* y *screen_name*, respectivamente. La preparación de estos datos es similar a la de la métrica anterior, pero sin el procesamiento previo, es decir, solamente se crean tres *dataframes* a partir del original que contienen las 10 entidades que más aparecen en la colección y sus respectivas frecuencias, introduciendo antes los datos en un diccionario de tipo *Counter*. Finalmente, las URLs más frecuentes requieren un paso más: la eliminación de dominios que se han utilizado para acortar otras webs y que, por tanto, no aportan ninguna información. La representación gráfica de estas tres métricas se ha realizado del mismo modo: utilizando un gráfico de barras horizontal. Estas figuras se han creado utilizando el objeto *Bar* de Plotly y muestran, como se puede observar en el ejemplo de la Figura 4.5, las entidades en el eje de ordenadas, dispuestas de mayor a menor frecuencia, y el número entero que indica la frecuencia en el eje de abscisas.

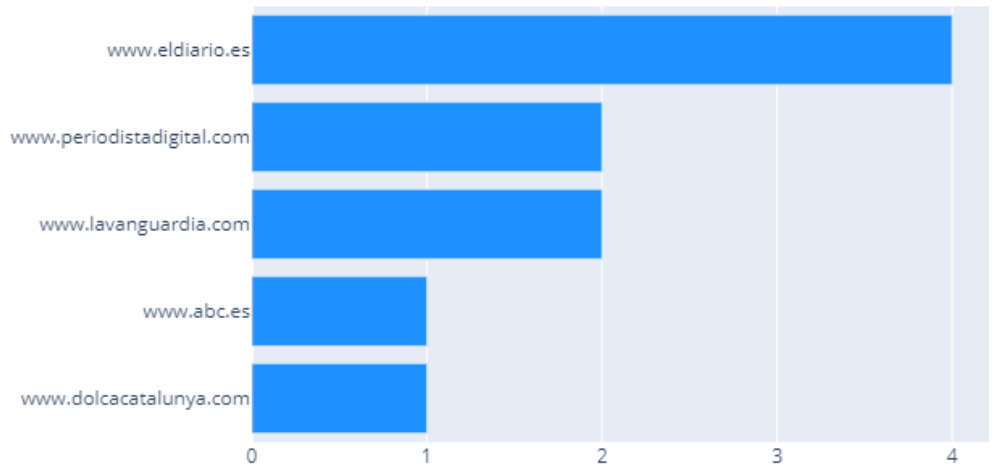

FIGURA 4.5: Visualización de URLs frecuentes

Análisis de emociones

Esta métrica permite conocer la carga emocional de un conjunto de tuits. Para ello, se ha utilizado el lexicón EmoLex, ya mencionado en la sección Análisis y visualización del capítulo anterior. Este corpus contiene, para cada palabra, un vector cuyos valores oscilan entre 0 y 1 y cuyas coordenadas corresponden a las ocho emociones básicas propuestas por Robert Plutchik en 1980 [35]: alegría, tristeza, anticipación, sorpresa, confianza, disgusto, enfado y miedo.

Dado el peor rendimiento obtenido utilizando la versión española del lexicón, antes del análisis se realiza un procesamiento previo de los datos que implica el uso de la biblioteca de código de Python *googletrans*, que traduce cada uno de los tuits, en caso de no estar escritos en inglés, para poder utilizar la versión original del corpus. A continuación, se filtran los datos utilizando una función auxiliar denominada *limpiar_tweet* que, a través de una expresión regular, elimina caracteres no deseados de todas las cadenas de texto. Finalmente, se separan todas las palabras y se pasan a minúsculas. Una vez terminado este procesamiento previo, se utiliza un bucle para extraer del lexicón el vector que contiene las emociones de cada palabra, en caso de que esté presente en el propio lexicón, y calcular el sentimiento final de toda la colección. Para obtener este sentimiento final, se calcula el sumatorio de cada una de las coordenadas de los vectores en un array de NumPy auxiliar del mismo tamaño. En este caso, la representación de los resultados de la métrica se muestran en un gráfico radial, como se puede ver en la Figura 4.6, haciendo uso del objeto *Scatterpolar* de Plotly e introduciendo directamente el vector con las «puntuaciones» finales de cada emoción.

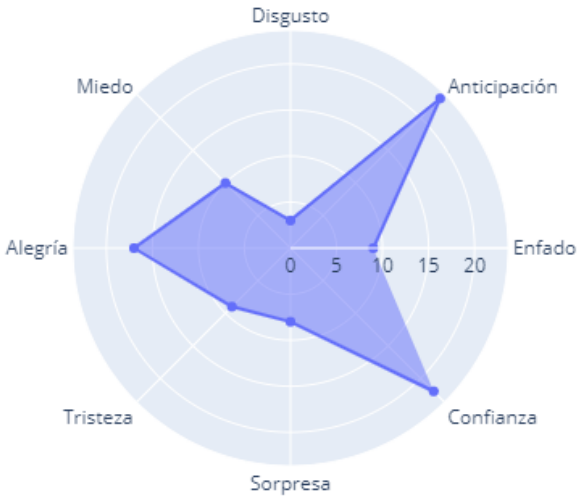

FIGURA 4.6: Visualización del análisis de emociones

Análisis de sentimiento

Con esta métrica se pretende, como en el análisis de emociones, averiguar y extraer información subjetiva de los tuits utilizando herramientas que permiten procesar lenguaje natural y analizar texto. Concretamente, se busca identificar la polaridad del conjunto de mensajes, es decir, el porcentaje de carga positiva o negativa que contiene la colección de mensajes relativa a una noticia. Para ello se pensó en recurrir, en un principio, a la biblioteca de código de Python Textblob [36]. Sin embargo, el lexicón EmoLex, ya utilizado para el análisis emocional, cuenta también con parámetros para medir la polaridad. La única diferencia en la implementación del análisis es que, del vector resultante tras el análisis emocional, se toman las dos primeras coordenadas. Estas contienen la «puntuación» de sentimiento positivo y negativo resultante de todos los tuits. Con estas cifras, y el número de tuits totales que tiene la colección, se puede averiguar el porcentaje de carga positiva, negativa y neutral total del conjunto. De este modo, utilizando el mismo vector de la métrica anterior, no es necesario volver a procesar los tuits y realizar cálculos adicionales.

Para la representación gráfica se ha utilizado el objeto *Indicator* de Plotly con formato *gauge*. Este medidor permite ver, como se puede ver en la Figura 4.7, la cantidad de carga positiva o negativa que tienen todos los mensajes recopilados, además de mostrar el porcentaje de carga positiva sobre la figura.

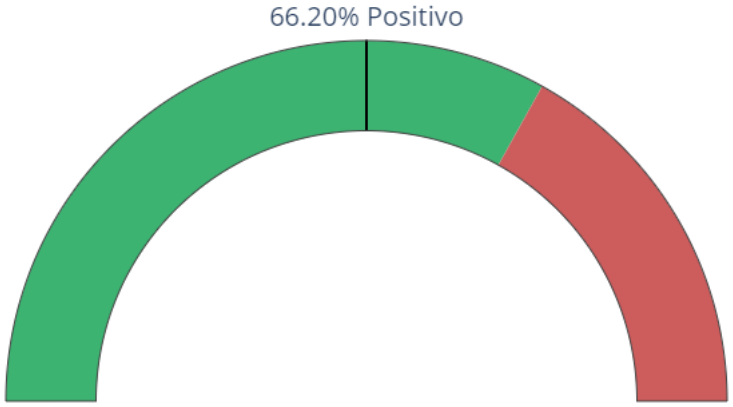

FIGURA 4.7: Visualización del análisis de sentimiento

Distribución de fuentes

Esta métrica no forma parte de los requisitos originales del proyecto, pero se decidió añadir en la fase de desarrollo de este porque requería una implementación sencilla que podía aportar información adicional que el usuario podría encontrar útil. La fuente de un tuit se puede encontrar en su atributo *source* y contiene una cadena de caracteres que siempre contiene la cadena «*Twitter for*» seguida de la aplicación o dispositivo empleado para publicar el mensaje en la red social. La columna del *dataframe* original que contiene esta información ya es procesada en el tratamiento inicial para eliminar la primera parte de la cadena y conservar, de este modo, la fuente únicamente. A continuación, se calcula el porcentaje de cada una de las fuentes dentro del conjunto y se representan los resultados en un gráfico de sectores, como se puede ver en la Figura 4.8.

Localización de los usuarios

El objetivo inicial de esta métrica era representar sobre un mapa las coordenadas geográficas asociadas a cada uno de los tuits analizados. Estas coordenadas se pueden encontrar en el atributo *coordinates* de un objeto tuit. Sin embargo, ninguno de los tuits analizados en las distintas pruebas realizadas durante el desarrollo de la aplicación contenían esta información, de hecho, es muy poco habitual que suceda. Debido a las limitaciones de la máquina en la que se ha desarrollado el proyecto, la cantidad de tuits de la colección no es lo suficientemente grande como para que pueda abarcar tuits que contengan la ubicación en la que han sido publicados. Por esta razón, se ha recurrido a otra alternativa para poder mostrar estas ubicaciones. El objeto *user*, atributo que incluyen todos los tuits, cuenta con un atributo denominado *location* que contiene, si ha sido rellenado por el usuario, una cadena de texto indicando la ubicación donde se encuentra. Por lo tanto, se ha seleccionado la biblioteca de Python GeoPy para obtener, a

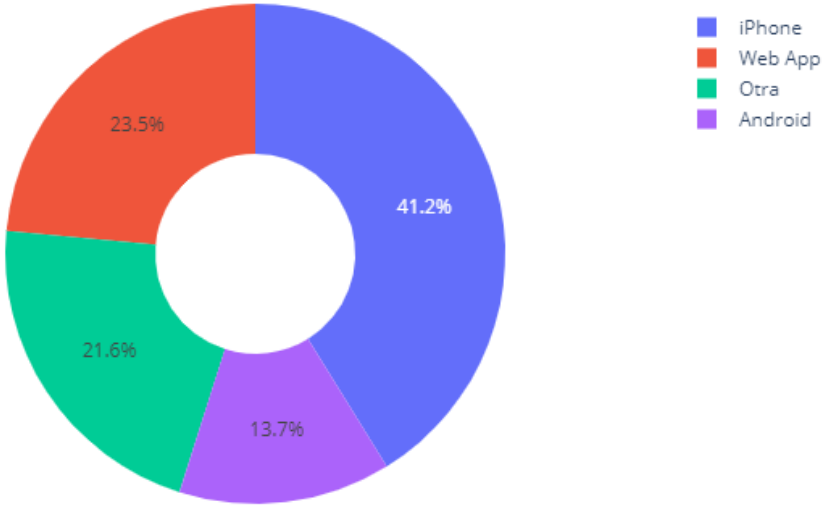

FIGURA 4.8: Visualización del análisis de fuentes

partir la columna que contiene este dato en el *dataframe* original, las coordenadas de estas ubicaciones y poder, finalmente, representarlas en un mapa.

Para esta representación se ha creado un objeto *Scattergeo* de Plotly, ya que permite mostrar directamente, a partir de dos listas que contengan las latitudes y las longitudes de todos los puntos, todas las ubicaciones en un mapa. Además, este mapa es dinámico, es decir, permite al usuario desplazarse por este, alejar o acercar la imagen y visualizar las coordenadas concretas de cada punto. Existen distintos *scopes* o tipos de mapas para representar estos puntos, pero en este trabajo se ha decidido utilizar un globo terráqueo por resultar más intuitivo y estético. Finalmente, sobre la figura se puede visualizar el porcentaje de usuarios cuya ubicación está especificada en su perfil, siempre que, además, pueda ser procesable por GeoPy. Se puede observar un ejemplo en la Figura 4.9.

Tuits destacados

Finalmente, la última métrica permite visualizar, de manera general, información general y adicional de la colección de tuits en una tabla interactiva. El contenido de esta tabla se genera a partir de la información general recopilada en el *dataframe* original. Sin embargo, solo se muestran las siguientes columnas: «Fecha», «Tweet», «Retweets», «Favoritos», «Usuario», «Seguidores», «Seguidos», «Verificado» y «Fuente». El único procesamiento previo es que los tuits que se muestran en la tabla están ordenados de mayor a menor número de retuits, de manera que el usuario pueda extraer, a primera vista, los mensajes más relevantes de todo el conjunto recopilado.

FIGURA 4.9: Visualización del análisis geográfico

Para crear la tabla se ha utilizado un objeto de tipo *Datatable*, proporcionado por la biblioteca *dash_table* de Dash. Inicialmente, la tabla solo muestra los 5 primeros tuits, pero esta opción es fácilmente modificable a través de un menús desplegable que se encuentra sobre la tabla y que permite cambiar este número entre los valores 5, 10, 20 y «Todos». La tabla es, como se ha dicho anteriormente, interactiva, es decir, cuenta con funcionalidad adicional que puede ser aprovechada con una entrada proporcionada por el usuario. Por ejemplo, el usuario puede ordenar de manera ascendente o descendente el contenido de la tabla haciendo clic sobre las flechas presentes en la columna que contenga el parámetro que se quiera ordenar. Además, el usuario puede también filtrar el contenido de la tabla (sobre el número de tuits que se estén mostrando en el momento del filtrado). Para ello, en la casilla que se encuentra debajo de la cabecera de cada columna que se quiera filtrar se pueden introducir los siguientes operadores:

- **=**: mostrará los valores iguales al valor introducido y sirve para cualquier tipo de contenido. Operador por defecto en las columnas con valores numéricos. Ejemplos: = 2020-05, = PabloIglesias, = 2000, 2000.
- **contains**: mostrará las celdas cuyas palabras o frases contengan el valor introducido. Operador por defecto en las columnas con texto. Ejemplos: contains Pablo, Pablo.
- **datestartswith**: mostrará las celdas cuyas fechas comiencen con el valor introducido, solo aplicable a las columnas que contengan una fecha (la primera, en este caso). Si se mete una fecha parcial, se tendrán en cuenta todas las fechas que contengan

esa parte. Ejemplos: datestartswith 2020-05-20 04:01:10, datestartswith 2020-05-20, datestartswith 2020-05.

- **>, <, >=, <=, !=**: mostrarán los valores mayores, menores, mayores o iguales, menores o iguales o distintos al valor introducido, respectivamente. Estos operadores sirven para cualquier tipo de contendido. Ejemplos: >55000, <= 2020-05-22, != Europa.

Con esta última métrica se completa, por lo tanto, la descripción del diseño y el desarrollo de la parte de análisis y visualización de este proyecto.

Capítulo 5

Resultados

En este capítulo se describirá el análisis de los resultados obtenidos tras completar el desarrollo de la aplicación propuesta en este trabajo. Este análisis contendrá la matriz de trazabilidad, las pruebas de validación y los resultados obtenidos en un cuestionario de satisfacción realizado por usuarios reales.

5.1. Matriz de trazabilidad

En esta sección se detallará la matriz de trazabilidad, es decir, se mostrarán, a través de una tabla, los requisitos planteados en la fase de análisis del proyecto y su relación con la aplicación final desarrollada (Tabla 5.1). De este modo, se puede comprobar si se ha cumplido con los objetivos planteados al comienzo del trabajo.

5.2. Pruebas de validación

En esta sección se analizarán distintos ejemplos de uso para verificar el correcto funcionamiento de la aplicación desarrollada, revisando que cumple con las especificaciones para revisar, de este modo, que logra su cometido. Se analizarán las dos partes del proyecto por separado, utilizando los mismos ejemplos en ambas secciones.

5.2.1. Recopilación de datos

Se han recopilado tuits relacionados con tres noticias diferentes realizando tres búsquedas: una el día 23 de junio y dos el 17 de julio de 2020 que retornaron 98, 43 y 111 mensajes, respectivamente. Como se ha explicado en la sección Recopilación de datos del capítulo anterior, debido al modo de búsqueda que se ha empleado, los resultados contienen una cantidad de tuits menor, pero de mayor relevancia. Las consultas que se hicieron fueron las siguientes:

TABLA 5.1: Matriz de trazabilidad

Requisito		Completado	Información adicional
RF1	RF1.1	✓	Consulta utilizando operadores de Twitter.
	RF1.2	✓	-
	RF1.3	✓	-
RF2	-	✓	-
RF3	-	✓	-
RF4	-	✓	Tablero web desplazable verticalmente.
RF5	RF5.1	✓	-
	RF5.2	✓	-
	RF5.3	✓	-
	RF5.4	✓	-
	RF5.5	✓	-
RF6	RF6.1	✓	Gráfico interactivo.
	RF6.2	✓	Gráfico interactivo.
RF7	RF7.1	✓	-
	RF7.2	✓	-
	RF7.3	✓	-
	RF7.4	✓	-
RF8	-	✓	-
RF9	-	✓	Gráfico interactivo.
RF10	RF10.1	✓	-
	RF10.2	✓	-
RF11	-	✓	-
RF12	-	✓	Gráfico interactivo.
RF13	RF13.1	✓	-
	RF13.2	✓	-
	RF13.3	✓	-
	RF13.4	✓	-
	RF13.5	✓	-
	RF13.6	✓	-
	RF13.7	✓	Añadidos fuente y si el usuario es verificado.
RF14	-	✓	Añadido análisis de fuentes.
RNF1	-	✓	-
RNF2	-	✓	-
RNF3	-	✓	-

- «correos OR to:Correos», es decir, todos los tuits que contuvieran la palabra «correos» o que estuvieran dirigidos a la cuenta @Correos [37].
- «funeral de Estado», es decir, todos los tuits que contuvieran la cadena «funeral de Estado» [38].
- «brote OR rebrote», es decir, todos los tuits que contuvieran las palabras «brote» o «rebrote» [39].

Se han hecho diversas pruebas y, con la implementación que se ha llevado a cabo, se ha conseguido obtener cientos de miles de mensajes en una sola colección, siempre que se utilicen los parámetros de búsqueda originales. La limitaciones existentes son la cantidad de tuits que pueda proporcionar la API de Twitter para una búsqueda concreta o la capacidad de procesamiento de la máquina que haga las peticiones. Si no se da ninguna de estas limitaciones, la búsqueda continúa, trabajando automáticamente entre

ventanas de tiempo, hasta que el usuario decida que debe terminar. De nuevo, se han modificado estos parámetros para poder desarrollar la parte de análisis y visualización y, de esta manera, comprobar las funcionalidades de la aplicación a menor escala con la máquina con la que se contaba.

Teniendo este factor en cuenta, las colecciones suelen rondar el centenar de tuits. Dependiendo de lo estricta que sea la consulta que introduzca el usuario, la respuesta incluirá menos mensajes o más ruido. Por ejemplo, la segunda consulta, además de ser una noticia reciente el día que fue realizada, es más estricta que las otras dos porque, por defecto, si no se introduce ningún operador, Twitter utiliza el operador AND. Por lo tanto, los tuits recopilados debían contener la cadena «funeral de Estado», no cualquiera de las tres palabras por separado. Por esta razón, la colección obtenida es bastante menor que en los otros dos casos.

Finalmente, dadas estas colecciones de tamaño manejable, se ha pasado a realizar la validación de la parte de análisis y visualización que, una vez implementada, es escalable y puede procesar colecciones de mayor tamaño siempre que se cuente con la capacidad de procesamiento suficiente.

5.2.2. Análisis y visualización

En esta sección se van a mostrar ejemplos de los resultados obtenidos tras el desarrollo de esta parte del proyecto. Primero, se puede ver en el Anexo B, y de manera general, la disposición final de todos los paneles en el tablero final, tras la ejecución de la aplicación analizando los resultados de la consulta «funeral de Estado». Aun así, se pueden ver ejemplos de los distintos tipos de gráficas con más detalle en la sección Métricas del capítulo anterior.

Cabe destacar que, normalmente, con colecciones de este tamaño, las métricas que muestran menos resultados son las que miden la frecuencia de entidades (*hashtags*, menciones y URLs) dentro del conjunto. Aun así, el funcionamiento es perfectamente válido y este número aumentaría utilizando colecciones de mayor tamaño, lo que significaría obtener resultados más completos y fiables. Sin embargo, llama la atención el buen funcionamiento del análisis emocional que, como se puede ver en la Figura 5.1, muestra una clara predominancia de tristeza sobre el resto de emociones en el análisis de la segunda noticia que se ha analizado, relacionada con el homenaje a las víctimas por el COVID-19 en España. Se puede observar otro ejemplo en la Figura 5.2, correspondiente a la búsqueda de tuits con la consulta «brote OR rebrote», realizada por el aumento reciente en el número de casos de pacientes de COVID-19), en la que se ve que el miedo es la emoción más acentuada. En ambos casos se observa una alta correlación entre las opiniones publicadas en la red social y el contenido de ambas noticias.

FIGURA 5.1: Análisis de emociones para la consulta «funeral de Estado»

Otro ejemplo a destacar tras el análisis de los resultados es la métrica que mide las URLs frecuentes. Aunque el resto de métricas que miden frecuencias, ya sean de palabras, *hashtags* o menciones, sean también métricas relevantes, tras diversas pruebas se ha llegado a la conclusión de que permiten obtener una visión más general que detallada de la información que se quiere extraer, es decir, son útiles para saber «qué se está comentando» a simple vista (la nube de palabras, especialmente). Sin embargo, en el caso de las URLs, los resultados que proporciona esta métrica suelen contener información más importante para el análisis de una noticia. Un ejemplo claro es el primer ejemplo propuesto en esta sección. Con la consulta «correos OR to:Correos» se pretendía obtener información sobre la campaña llevada a cabo por Correos con motivo del orgullo LGTBI el pasado junio de 2020, ya que había sido tanto aplaudida como criticada en las redes. Sin embargo, aunque la consulta en sí no fuera muy estricta, dando pie, por tanto, a que hubiera ruido en los resultados, se puede ver en la Figura 5.3 que el dominio que más aparece es la página web de El Español, diario que publicó el análisis de los gastos y los beneficios de dicha campaña que citaban la mayoría de usuarios que se posicionaban a favor de la campaña, es decir, que, en caso de conocer la fuente original de una noticia, a través del análisis del impacto en Twitter se puede llegar a identificar el medio original que ha publicado la noticia, además de los más populares, algo que, en definitiva, puede ser de gran utilidad.

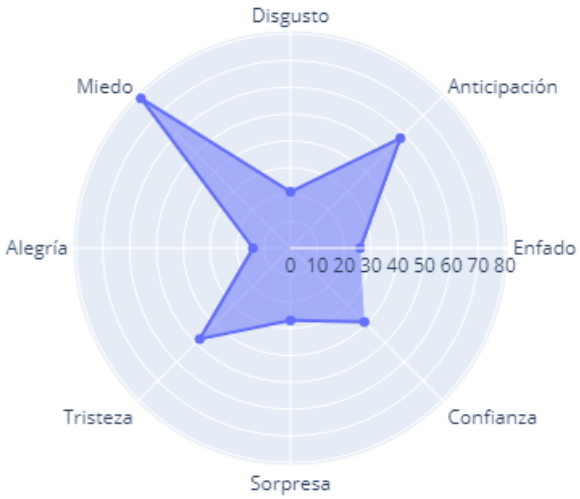

FIGURA 5.2: Análisis de emociones para la consulta «brote OR rebrote»

5.3. Cuestionario de satisfacción

En esta sección se detallará cómo se ha abordado el objetivo de evaluar el grado de satisfacción de los usuarios. Para ello, se ha decidido utilizar el modelo GQM (*Goal-Question-Metric*) [40]. Este modelo se caracteriza por dividirse en tres niveles:

- **Nivel conceptual:** se definen los objetivos (*goals*) que se quieren medir teniendo en cuenta distintos puntos de vista. En este caso, serán dos: el grado de utilidad que tienen cada una de las métricas y el grado de satisfacción con sus representaciones gráficas.
- **Nivel operacional:** se elabora un conjunto de preguntas para los objetivos que se quieren estudiar, con el fin de establecer cuál es la meta de un objetivo concreto. En este caso, se ha decidido utilizar una pregunta por *goal*:
  - «Me ha resultado útil esta métrica».
  - «Estoy satisfecho con la representación gráfica».
- **Nivel cuantitativo:** se define un conjunto de métricas asociadas a cada pregunta para que las respuestas que se introduzcan puedan ser medibles. En este caso, las preguntas definidas en el nivel operacional cuentan la misma y única métrica: la escala de Likert, propuesta por Rensis Likert en 1932 [41]. Esta métrica es la más utilizada en encuestas de investigación y se caracteriza por dividir la respuesta en 5 niveles:

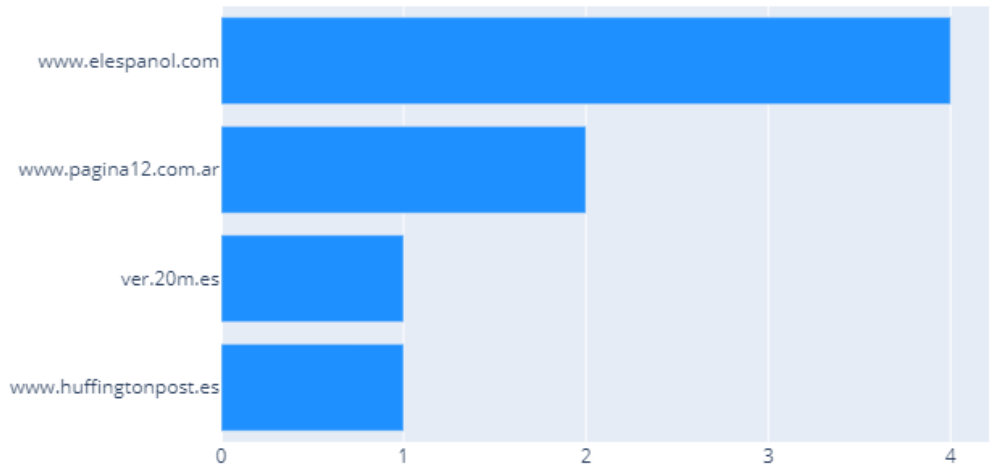

FIGURA 5.3: URLs frecuentes para la consulta «correos OR to:Correos»

1. Totalmente en desacuerdo
2. En desacuerdo
3. Ni de acuerdo ni en desacuerdo
4. De acuerdo
5. Totalmente de acuerdo

Una vez definidos todos los elementos del modelo, se crea el formulario de satisfacción. La plataforma utilizada para ello ha sido Google Forms, por su accesibilidad y facilidad de configuración. Además, para que el usuario examine antes todos los elementos de la aplicación que se quieren medir, se ha elaborado un guion que describe los pasos a seguir en una sesión. Este guion se puede ver en el Anexo C.

Este cuestionario ha sido realizado por 7 personas y los resultados de las respuestas se pueden observar detalladamente en el Anexo D. A continuación, en la Figura 5.4, se puede ver un resumen de las medias de las puntuaciones obtenidas, en cada una de de las preguntas, por cada métrica.

Se ha añadido también la media total de utilidad y de satisfacción con la representación gráfica de todas las métricas. Como se puede comprobar, en ambos casos la media supera los 4 puntos, por lo que se podría decir que se ha cumplido con el objetivo de evaluar el grado de satisfacción de los usuarios con la aplicación resultante y que, además, los resultados han sido razonablemente positivos. Sin embargo, cabe destacar que tanto

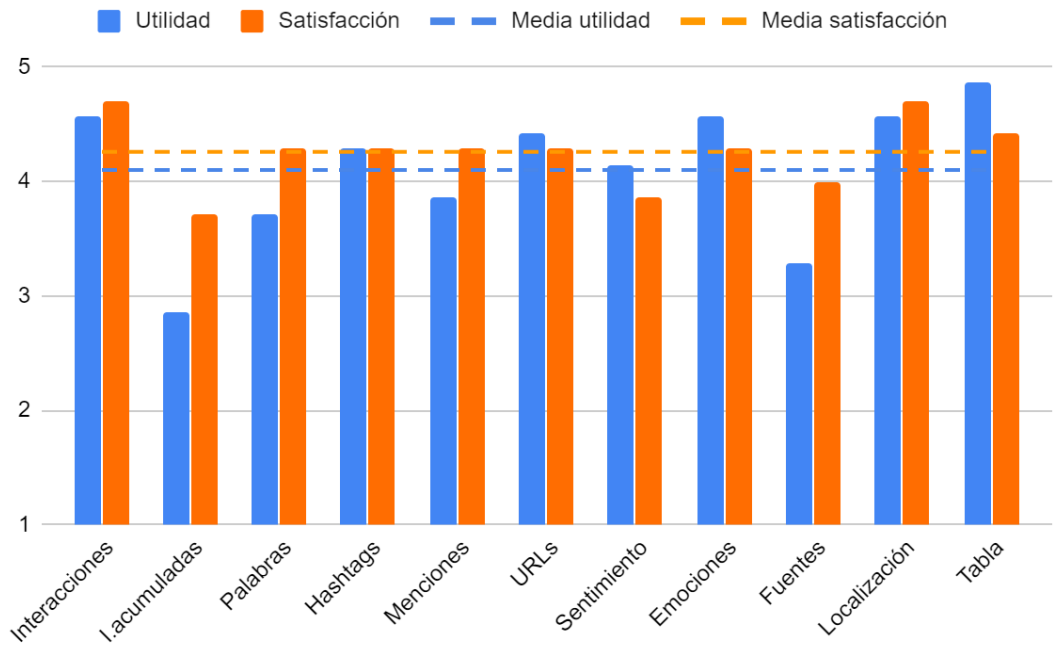

FIGURA 5.4: Puntuación media por métrica

la utilidad como la satisfacción con la representación de la métrica que mide las interacciones acumuladas a lo largo del tiempo han sido claramente menores que el resto de métricas. Además, de la utilidad de la distribución de fuentes desde las que se han enviado los tuits y la satisfacción con la representación del análisis de sentimiento positivo o negativo.

Capítulo 6

Conclusiones y trabajo futuro

Para terminar con esta memoria, en este capítulo se expondrán las conclusiones a las que se ha llegado tras la realización del trabajo y las ideas para mejorar la aplicación desarrollada en un futuro.

6.1. Conclusiones

Para concluir, se puede confirmar que este proyecto ha cumplido con los objetivos planteados al comienzo de este. A pesar de ser un propuesta nueva, cuya viabilidad se desconocía en un comienzo, no solo se ha desarrollado una herramienta que permite a un usuario, realizar un análisis, y visualizar los resultados de este, de un conjunto de mensajes enviados a través de la red social Twitter de la manera más objetiva posible, sino que se ha conseguido un notable grado de satisfacción por parte de usuarios reales que han probado la aplicación.

Es importante ser consciente de la ventaja que puede suponer una aplicación de este tipo en la actualidad, cuando la desinformación es uno de los problemas a los que se enfrenta la sociedad casi diariamente. Conocer el impacto de una noticia, ya sea falsa o verdadera, en el medio en el que se propaga más rápido puede resultar de gran utilidad, y este proyecto, aun con limitaciones, consigue justo eso; es esa la principal motivación que ha llevado a terminar el trabajo con éxito. A pesar de esto, todavía existen maneras para mejorar el rendimiento general de la herramienta, tanto depurando los puntos flacos como añadiendo funcionalidades nuevas.

Para finalizar, a modo de reflexión personal, el desarrollo de esta aplicación, y, en definitiva, enfrentarse a este trabajo, ha supuesto un verdadero reto. Un proyecto de este calibre ha requerido poner en práctica y aunar casi todos los conocimientos adquiridos a lo largo de los últimos años, además de, a través de metodologías también aprendidas en el grado, llevar a cabo un ejercicio de disciplina y organización sin el que no habría sido posible terminarlo.

6.2. Trabajo futuro

A lo largo del desarrollo del proyecto se ha contemplado la inclusión de numerosas mejoras que, inicialmente, no constaban en la planificación inicial del trabajo. Además, gracias a la experiencia de usuarios reales con la aplicación, que han aportado puntos de vista diferentes, se han podido identificar nuevos cambios que, en definitiva, pueden mejorar la herramienta. A continuación, se muestran algunas de estas futuras mejoras:

- Mejoras técnicas que no llegaron a incluirse en la implementación de algunas de las métricas como, por ejemplo, modificar el análisis geográfico para que trabaje sobre los tuits publicados y no sobre los usuarios, ya que eso ahorraría tiempo de ejecución, o la lematización de palabras cuando se analiza la frecuencia de estas dentro de la colección de tuits o se realizan los análisis de sentimiento y emociones.
- Mejoras gráficas en aquellas métricas cuyo grado de satisfacción entre los usuarios, tanto con la utilidad como con la representación gráfica, ha resultado estar a menor nivel tras el cuestionario realizado en la fase de análisis de resultados del proyecto.
- El despliegue de la aplicación en un servidor, de manera que se pueda acceder a los resultados del análisis remotamente a través de un navegador web.
- Una vez desplegada la aplicación, hacer las modificaciones pertinentes para que exista la opción de hacer uso de la búsqueda de tuits a través de un flujo en tiempo real. De este modo, se actualizaría dinámicamente la información visualizada en la aplicación pudiendo, así, monitorizar el impacto de una noticia en Twitter en tiempo real.
- Finalmente, la inclusión de más redes sociales o métricas a la herramienta para que el análisis sea del todo completo.

Bibliografía

[1] B. Martens, L. Aguiar, E. Gómez-Herrera y F. Müller-Langer. «The Digital Transformation of News Media and the Rise of Disinformation and Fake News». En: *SSRN Electronic Journal* (2018). DOI: 10.2139/ssrn.3164170.

[2] Twitter. *Tweet object — Twitter Developers*. 2020. URL: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object (visitado 17-07-2020).

[3] Twitter. *User object — Twitter Developers*. 2020. URL: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/user-object (visitado 17-07-2020).

[4] Twitter. *Entities object — Twitter Developers*. 2020. URL: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/entities-object (visitado 17-07-2020).

[5] G2.com. *Best Social Media Monitoring Software in 2020 | G2*. 2020. URL: https://www.g2.com/categories/social-media-monitoring?order=popular (visitado 17-07-2020).

[6] Hootsuite. *Social media listening with Hootsuite Insights Powered by Brandwatch*. 2020. URL: https://hootsuite.com/es/productos/insights (visitado 17-07-2020).

[7] Brandwatch. *Brandwatch*. 2020. URL: https://www.brandwatch.com/es/ (visitado 17-07-2020).

[8] Brandwatch. *Brandwatch Vizia: Social and Marketing Data Visualizations | Brandwatch*. 2020. URL: https://www.brandwatch.com/products/vizia/ (visitado 17-07-2020).

[9] Sprout Social. *Sprout Social: Social Media Management Solutions*. 2020. URL: https://sproutsocial.com/ (visitado 17-07-2020).

[10] Sprout Social. *Social Media Listening | Sprout Social*. 2020. URL: https://sproutsocial.com/features/social-media-listening/ (visitado 17-07-2020).

[11] Union Metrics. *How Far Did Your Tweets Travel? | TweetReach*. 2020. URL: https://tweetreach.com/ (visitado 17-07-2020).

[12] StatCounter. *Desktop Operating System Market Share Worldwide | StatCounter Global Stats*. 2020. URL: https://gs.statcounter.com/os-market-share/desktop/worldwide (visitado 17-07-2020).

[13] JetBrains s.r.o. *PyCharm: el IDE de Python para desarrolladores profesionales, por JetBrains*. 2020. URL: https://www.jetbrains.com/es-es/pycharm/ (visitado 17-07-2020).

[14] *Sign in · GitLab*. URL: https://git.eps.uam.es/users/sign_in (visitado 17-07-2020).

[15] *Formularios de Google: crea y analiza encuestas de forma gratuita*. URL: https://www.google.es/intl/es/forms/about/ (visitado 17-07-2020).

[16] J. Roesslein. *Tweepy Documentation — tweepy 3.8.0 documentation*. 2020. URL: http://docs.tweepy.org/en/latest/ (visitado 17-07-2020).

[17] M. Pavloski. *Accessing the Twitter API with Python*. URL: https://stackabuse.com/accessing-the-twitter-api-with-python/ (visitado 17-07-2020).

[18] D. Aguilar. *jsonpickle Documentation — jsonpickle 1.3 documentation*. 2016. URL: https://jsonpickle.readthedocs.io/en/latest/ (visitado 17-07-2020).

[19] The pandas development team. *pandas - Python Data Analysis Library*. 2020. URL: https://pandas.pydata.org/ (visitado 17-07-2020).

[20] NumPy. *NumPy*. 2020. URL: https://numpy.org/ (visitado 17-07-2020).

[21] The Matplotlib development team. *Matplotlib: Python plotting — Matplotlib 3.2.2 documentation*. 2020. URL: https://matplotlib.org/index.html (visitado 17-07-2020).

[22] Plotly. *Plotly Python Graphing Library | Python | Plotly*. 2020. URL: https://plotly.com/python/ (visitado 17-07-2020).

[23] *Dash Documentation & User Guide | Plotly*. URL: https://dash.plotly.com/ (visitado 17-07-2020).

[24] A. C. Müller. *WordCloud for Python documentation — wordcloud 1.6.0.post88+ge196d19 documentation*. 2020. URL: https://amueller.github.io/word_cloud/ (visitado 17-07-2020).

[25] NLTK Project. *Natural Language Toolkit — NLTK 3.7 documentation*. 2020. URL: https://www.nltk.org/ (visitado 17-07-2020).

[26] S. M. Mohammad y P. D. Turney. «Crowdsourcing a Word-Emotion Association Lexicon». En: *CoRR* abs/1308.6297 (2013). arXiv: 1308.6297.

[27] SuHun Han. *Googletrans: Free and Unlimited Google translate API for Python — Googletrans 2.4.0 documentation*. 2018. URL: https://py-googletrans.readthedocs.io/en/latest/ (visitado 17-07-2020).

[28] GeoPy Contributors. *Welcome to GeoPy's documentation! — GeoPy 2.0.0 documentation*. 2018. URL: https://geopy.readthedocs.io/en/stable/ (visitado 17-07-2020).

[29] Twitter. *Overview — Twitter Developers*. 2020. URL: https://developer.twitter.com/en/docs/basics/authentication/overview (visitado 17-07-2020).

[30] Twitter. *Application-only authentication — Twitter Developers*. 2020. URL: https://developer.twitter.com/en/docs/basics/authentication/oauth-2-0/application-only (visitado 17-07-2020).

[31] Twitter. *Standard search — Twitter Developers*. 2020. URL: https://developer.twitter.com/en/docs/tweets/search/overview/standard (visitado 17-07-2020).

[32] Twitter. *statuses/filter — Twitter Developers*. 2020. URL: https://developer.twitter.com/en/docs/tweets/filter-realtime/overview/statuses-filter (visitado 17-07-2020).

[33] Twitter. *Standard operators — Twitter Developers*. 2020. URL: https://developer.twitter.com/en/docs/tweets/rules-and-filtering/overview/standard-operators (visitado 17-07-2020).

[34] *JSON*. URL: https://www.json.org/json-en.html (visitado 17-07-2020).

[35] R. Plutchik. «A General Psychoevolutionary Theory of Emotion». En: *Theories of Emotion*. Ed. por R. Plutchik y H. Kellerman. Academic Press, 1980. Cap. 1, págs. 3-33. ISBN: 9780125587013. DOI: 10.1016/B978-0-12-558701-3.50007-7.

[36] S. Loria. *TextBlob: Simplified Text Processing — TextBlob 0.16.0 documentation*. 2020. URL: https://textblob.readthedocs.io/en/dev/ (visitado 17-07-2020).

[37] A. Criado. *El arcoíris de Correos: así convirtió una inversión de 12.500 € en un impacto de medio millón en tres días*. 2020. URL: https://www.elespanol.com/invertia/empresas/20200622/arcoiris-correos-convirtio-inversion-eur-impacto-millon/499450392_0.html (visitado 17-07-2020).

[38] La Vanguardia. *El funeral de Estado por las víctimas de la Covid, en imágenes*. 2020. URL: https://www.lavanguardia.com/politica/20200716/482331288131/funeral-de-estado-victimas-coronavirus-espana-imagenes.html (visitado 17-07-2020).

[39] Agencia EFE. *Las autonomías ensayan cómo frenar la escalada de contagios sin confinamiento*. 2020. URL: https://www.efe.com/efe/espana/portada/las-autonomias-ensayan-como-frenar-la-escalada-de-contagios-sin-confinamiento/10010-4299546 (visitado 17-07-2020).

[40] R. van Solingen (Revision), V. Basili (Original article 1994 ed.), G. Caldiera (Original article 1994 ed.) y H. D. Rombach (Original article 1994 ed.) «Goal Question Metric (GQM) Approach». En: *Encyclopedia of Software Engineering*. American Cancer Society, 2002. ISBN: 9780471028956. DOI: 10.1002/0471028959.sof142.

[41] R. Likert. «A Technique for the Measurement of Attitudes». En: *Archives of psychology* (1932).

Anexos

Anexo A

Tweet object


```
{
  "created_at": "Thu Jun 18 07:30:00 +0000 2020",
  "id": 1273518308503908400,
  "id_str": "1273518308503908353",
  "full_text": "En Correos somos amarillos y nos encanta que
      cada uno pueda ser el color que quiera. Por eso lo
     celebramos con este sello, que no es solo amarillo. #
     NoSoloAmarillo #Orgullo2020",
  "truncated": false,
  "display_text_range": [
    0,
    176
  ],
  "entities": {
    "hashtags": [
      {
        "text": "NoSoloAmarillo",
        "indices": [
          148,
          163
        ]
      },
      {
        "text": "Orgullo2020",
        "indices": [
          164,
          176
        ]
      }
    ],
    "symbols": [],
```

```
    "user_mentions": [],
    "urls": []
  },
  "metadata": {
    "result_type": "popular",
    "iso_language_code": "es"
  },
  "source": "<a href=\"https://ads-api.twitter.com\" rel=\"
     nofollow\">Twitter for Advertisers</a>",
  "in_reply_to_status_id": null,
  "in_reply_to_status_id_str": null,
  "in_reply_to_user_id": null,
  "in_reply_to_user_id_str": null,
  "in_reply_to_screen_name": null,
  "user": {
    "id": 198535580,
    "id_str": "198535580",
    "name": "Correos",
    "screen_name": "Correos",
    "location": "Madrid (España)",
    "description": "¡Bienvenidos a nuestro canal de Twitter!
       \nPara cualquier consulta, te atenderemos en
      @CorreosAtiende.\nPolítica de privacidad: https://t.
      co/on4vL3iCEh",
    "url": "https://t.co/Lmj8h5LAkP",
    "entities": {
      "url": {
        "urls": [
          {
            "url": "https://t.co/Lmj8h5LAkP",
            "expanded_url": "http://www.correos.es",
            "display_url": "correos.es",
            "indices": [
              0,
              23
            ]
          }
        ]
      },
      "description": {
        "urls": [
          {
            "url": "https://t.co/on4vL3iCEh",
```

```
            "expanded_url": "http://bit.ly/36bS5XN",
            "display_url": "bit.ly/36bS5XN",
            "indices": [
              126,
              149
            ]
          }
        ]
      }
    },
    "protected": false,
    "followers_count": 51738,
    "friends_count": 265,
    "listed_count": 497,
    "created_at": "Mon Oct 04 15:37:29 +0000 2010",
    "favourites_count": 2984,
    "utc_offset": null,
    "time_zone": null,
    "geo_enabled": true,
    "verified": true,
    "statuses_count": 113763,
    "lang": null,
    "contributors_enabled": false,
    "is_translator": false,
    "is_translation_enabled": false,
    "profile_background_color": "08426B",
    "profile_background_image_url": "http://abs.twimg.com/
      images/themes/theme16/bg.gif",
    "profile_background_image_url_https": "https://abs.twimg
      .com/images/themes/theme16/bg.gif",
    "profile_background_tile": false,
    "profile_image_url": "http://pbs.twimg.com/
      profile_images/1273500341653385217/tycn5nVP_normal.
      jpg",
    "profile_image_url_https": "https://pbs.twimg.com/
      profile_images/1273500341653385217/tycn5nVP_normal.
      jpg",
    "profile_banner_url": "https://pbs.twimg.com/
      profile_banners/198535580/1592461117",
    "profile_link_color": "1628F2",
    "profile_sidebar_border_color": "FFFFFF",
    "profile_sidebar_fill_color": "FAFAFA",
    "profile_text_color": "000000",
```

```
    "profile_use_background_image": true,
    "has_extended_profile": false,
    "default_profile": false,
    "default_profile_image": false,
    "following": null,
    "follow_request_sent": null,
    "notifications": null,
    "translator_type": "none"
  },
  "geo": null,
  "coordinates": null,
  "place": null,
  "contributors": null,
  "is_quote_status": false,
  "retweet_count": 3696,
  "favorite_count": 17112,
  "favorited": false,
  "retweeted": false,
  "lang": "es"
}
```

Anexo B

Ejemplo de ejecución

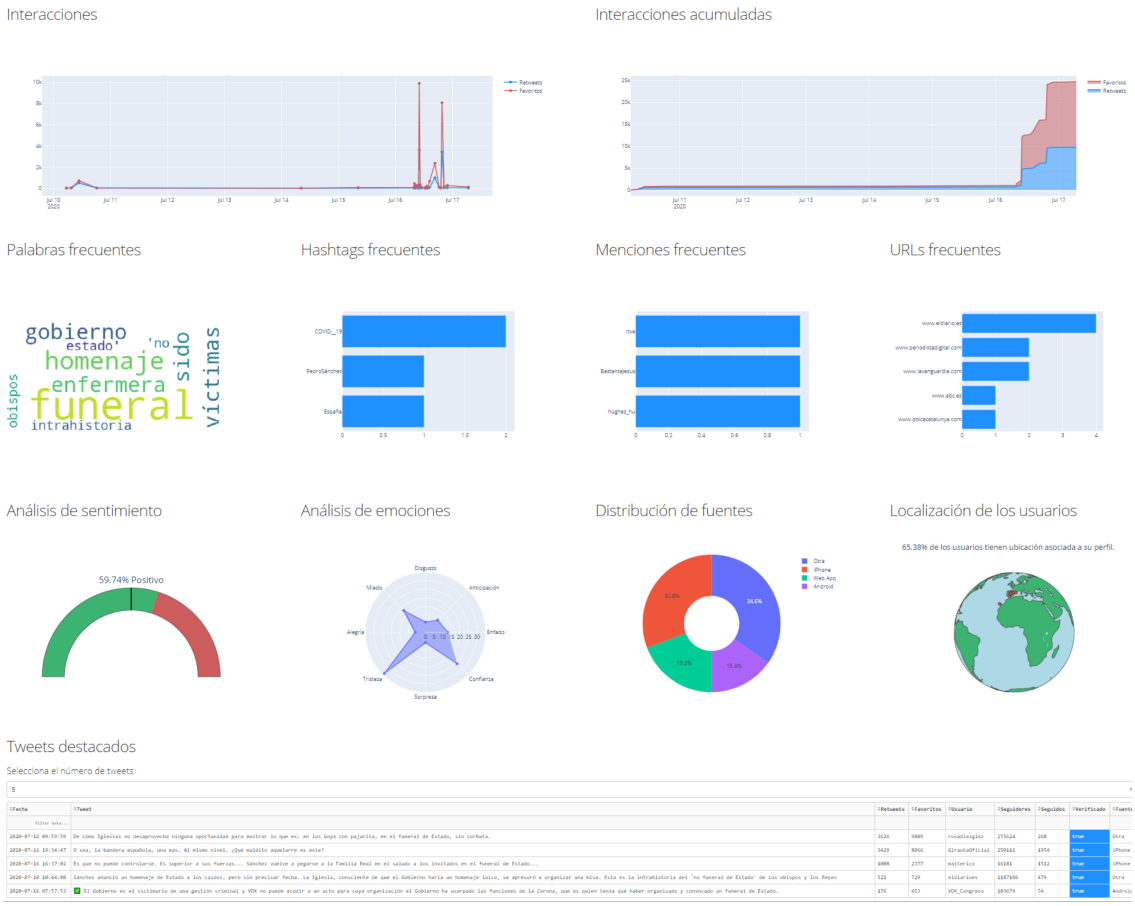

FIGURA B.1: Visión general del tablero completo

Anexo C

Cuestionario de satisfacción - Pasos a seguir

1. En la primera fila del panel se pueden observar dos gráficas que muestran la variación de interacciones («Retuits» y «Favoritos»), no acumuladas y acumuladas, a lo largo del tiempo. Puede acercar la imagen en un gráfico seleccionando una región dentro de este con el cursor. Para volver a la disposición inicial puede hacer clic dos veces sobre el gráfico. Además, si pasa el cursor por encima de cualquier punto, podrá visualizar el texto que contiene el tuit.

2. A continuación, en la siguiente fila, puede observar cuatro gráficas que muestran las palabras, los hashtags, las menciones y los dominios más frecuentes dentro de la colección de tuits analizada. Menos en la nube de palabras, puede acercar la imagen en un gráfico seleccionando una región dentro de este con el cursor. Para volver a la disposición inicial puede hacer clic dos veces sobre el gráfico. Además, si pasa el cursor por encima de cualquier punto, podrá visualizar la frecuencia de un elemento.

3. Justo debajo, la siguiente sección comienza dos gráficas que permiten visualizar en análisis de sentimientos que se ha hecho sobre el texto de los tuits. Se muestra tanto el análisis de sentimiento positivo o negativo (con un indicador y el porcentaje positivo) como el análisis de emociones (con un gráfico radial). Puede acercar la imagen en este último seleccionando una región dentro del gráfico con el cursor. Para volver a la disposición inicial puede hacer clic dos veces sobre el gráfico.

4. A continuación, se muestra un gráfico circular que muestra cómo se dividen las fuentes desde las que se han enviado los tuits de la colección. Arriba a la derecha hay una leyenda que indica a qué fuente corresponde cada sector.

5. Finalmente, esta sección del panel termina con un globo terráqueo que muestra con puntos rojos la localización de los usuarios que han enviado los tuits, siempre que estos muestren su ubicación en la descripción del perfil (encima si indicará el porcentaje que lo hace). Puede acercar la imagen en el gráfico utilizando la rueda de su ratón y moverse por el globo haciendo clic y arrastrando el cursor sobre este. Para

volver a la disposición inicial puede hacer clic dos veces sobre el gráfico. Además, si pasa el cursor por encima de cualquier punto, podrá visualizar las coordenadas de este.

6. Por último, se muestra una tabla con los cinco tuits más retuiteados junto con información adicional sobre cada uno de ellos. Sin embargo, puede elegir el número de tuits que se visualizan con el menú desplegable que se encuentra sobre la tabla. Puede ordenar de manera ascendente o descendente haciendo clic sobre las flechas presentes en la columna que contenga el parámetro que deseemos ordenar. Puede también filtrar el contenido de la tabla en la casilla que se encuentra debajo de la cabecera de cada columna. Para ello, existen diversos operadores:

   - **=**: mostrará los valores iguales al valor introducido y sirve para cualquier tipo de contenido. Operador por defecto en las columnas con valores numéricos. Ejemplos: = 2020-05, = PabloIglesias, = 2000, 2000.
   - **contains**: mostrará las celdas cuyas palabras o frases contengan el valor introducido. Operador por defecto en las columnas con texto. Ejemplos: contains Pablo, Pablo.
   - **datestartswith**: mostrará las celdas cuyas fechas comiencen con el valor introducido, solo aplicable a las columnas que contengan una fecha (la primera, en este caso). Si se mete una fecha parcial, se tendrán en cuenta todas las fechas que contengan esa parte. Ejemplos: datestartswith 2020-05-20 04:01:10, datestartswith 2020-05-20, datestartswith 2020-05.
   - **>, <, >=, <=, !=**: mostrarán los valores mayores, menores, mayores o iguales, menores o iguales o distintos al valor introducido, respectivamente. Estos operadores sirven para cualquier tipo de contendido. Ejemplos: >55000, <= 2020-05-22, != Europa.

   Una vez introducido el operador y el valor que se quieren utilizar para filtrar la tabla, pulse la tecla «Enter» para hacerlo.

7. Ahora que ha terminado de examinar las distintas métricas, y sus representaciones gráficas, que se han tenido en el análisis; por favor, haga clic en el siguiente enlace: `https://forms.gle/KJNA3iiBdyCbbWEk8` y complete el cuestionario de satisfacción.

Anexo D

Cuestionario de satisfacción - Respuestas

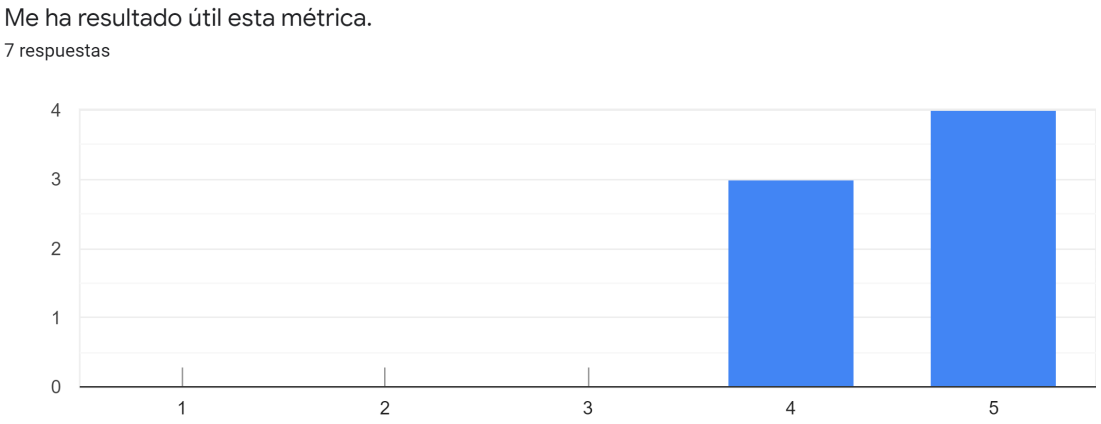

FIGURA D.1: Resumen de respuestas para la métrica «Interacciones»

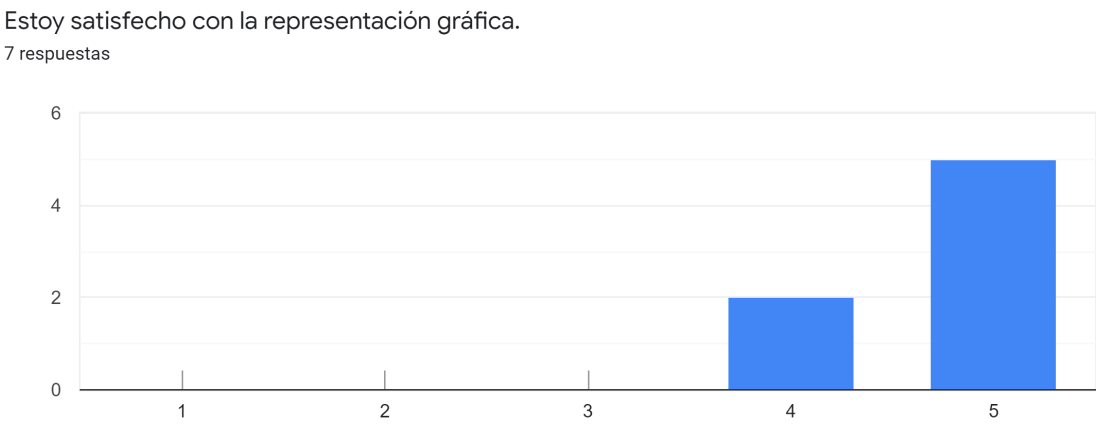

FIGURA D.2: Resumen de respuestas para la métrica «Interacciones»

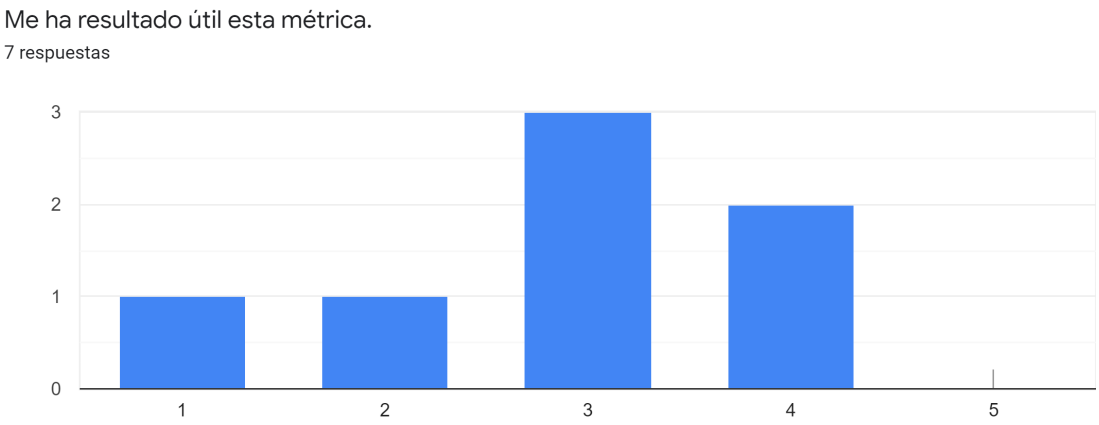

FIGURA D.3: Resumen de respuestas para la métrica «Interacciones acumuladas»

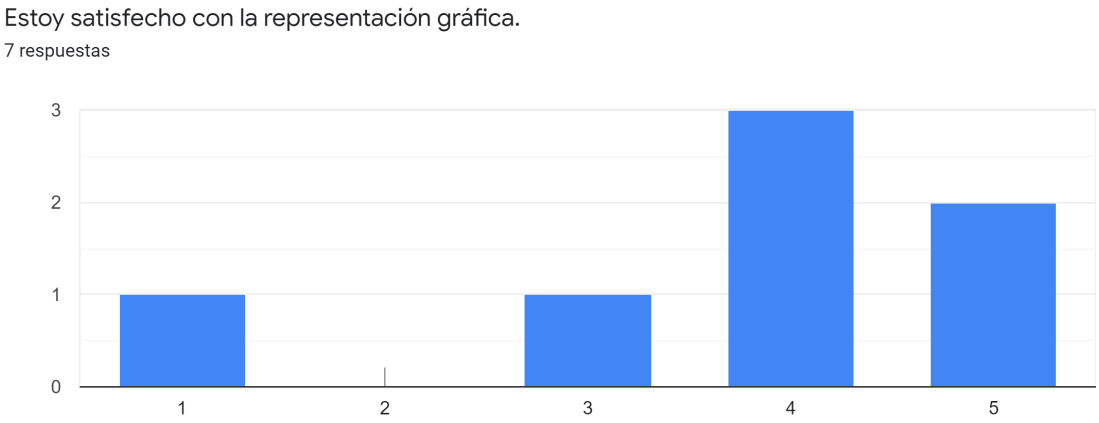

FIGURA D.4: Resumen de respuestas para la métrica «Interacciones acumuladas»

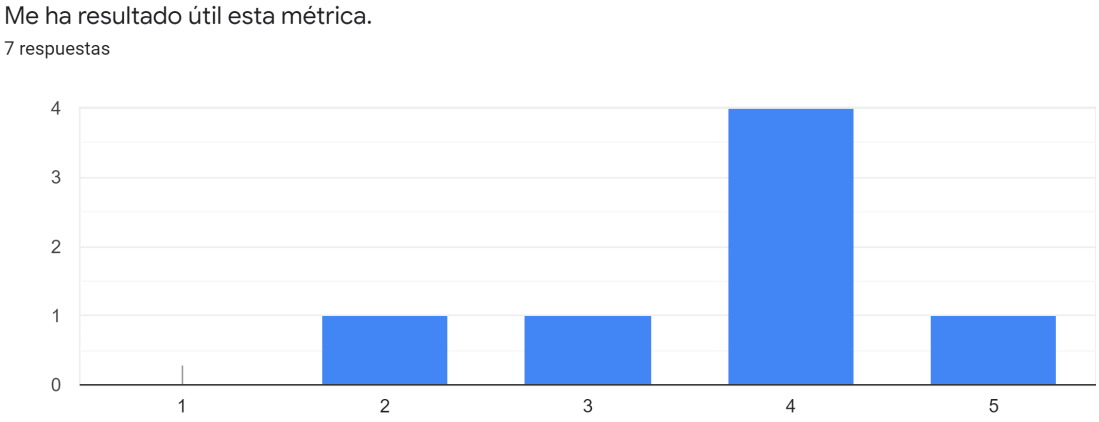

FIGURA D.5: Resumen de respuestas para la métrica «Palabras frecuentes»

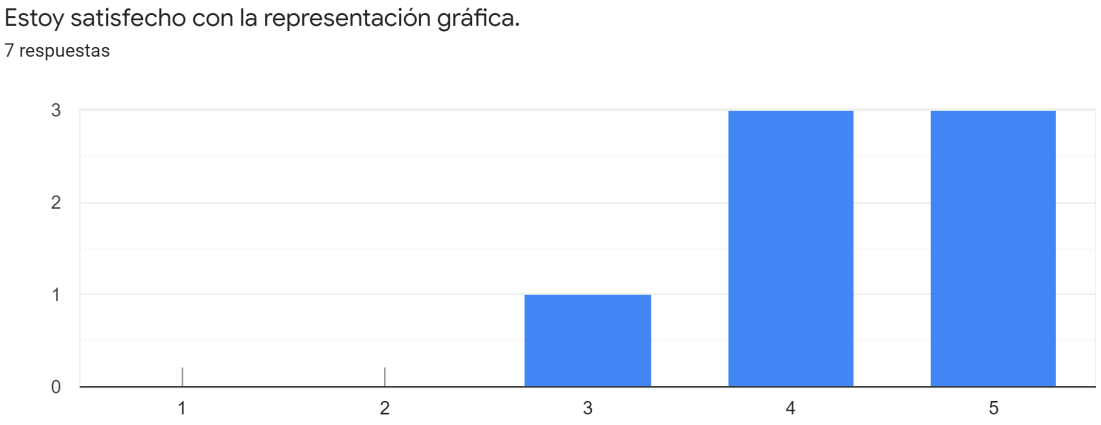

FIGURA D.6: Resumen de respuestas para la métrica «Palabras frecuentes»

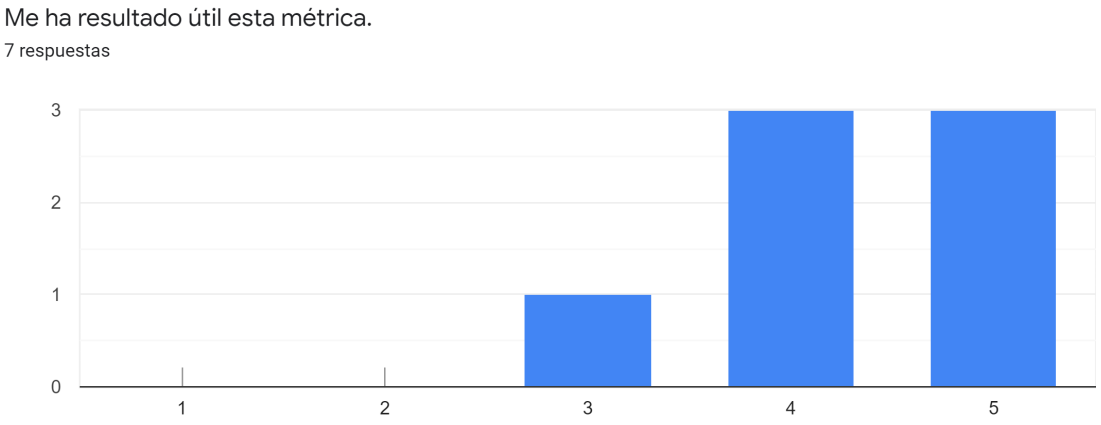

FIGURA D.7: Resumen de respuestas para la métrica «*Hashtags* frecuentes»

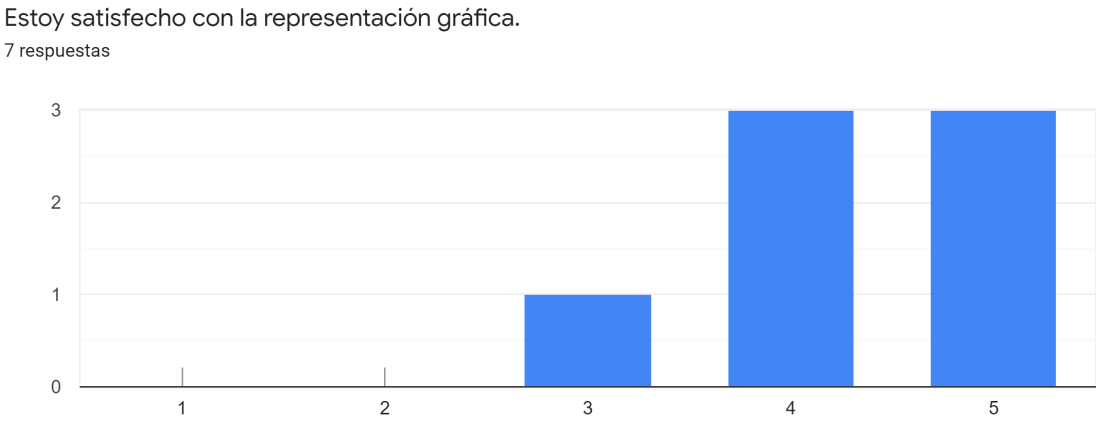

FIGURA D.8: Resumen de respuestas para la métrica «*Hashtags* frecuentes»

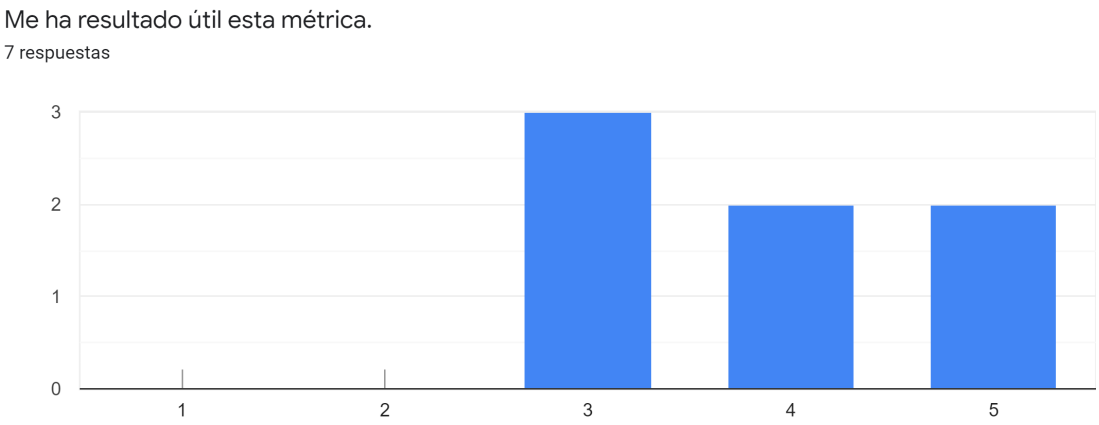

FIGURA D.9: Resumen de respuestas para la métrica «Menciones frecuentes»

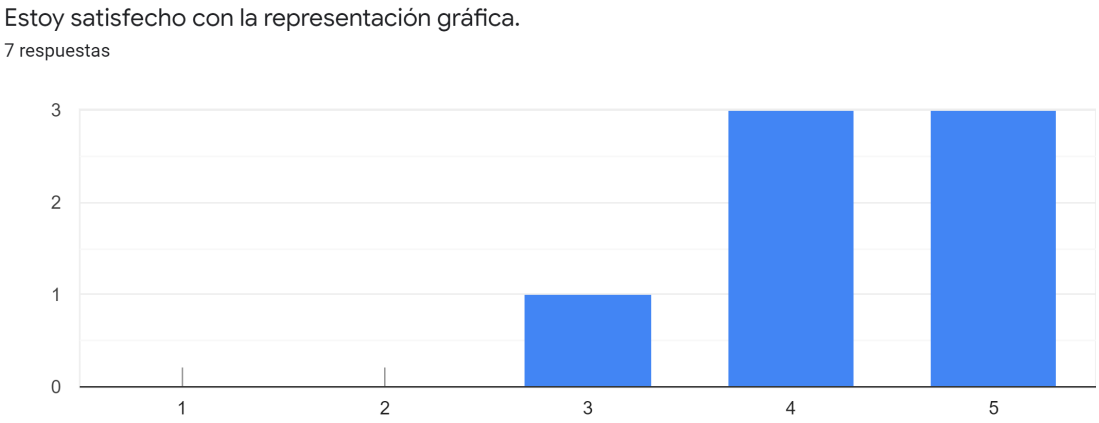

FIGURA D.10: Resumen de respuestas para la métrica «Menciones frecuentes»

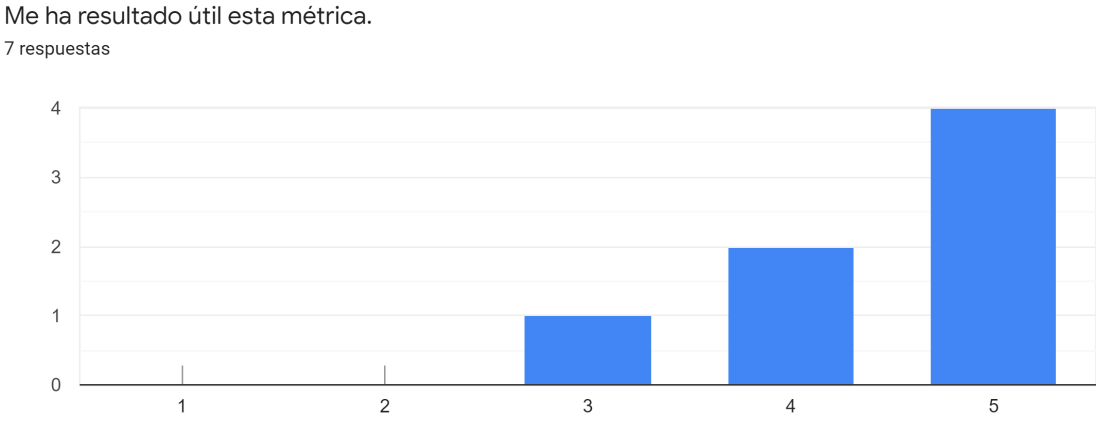

FIGURA D.11: Resumen de respuestas para la métrica «URLs frecuentes»

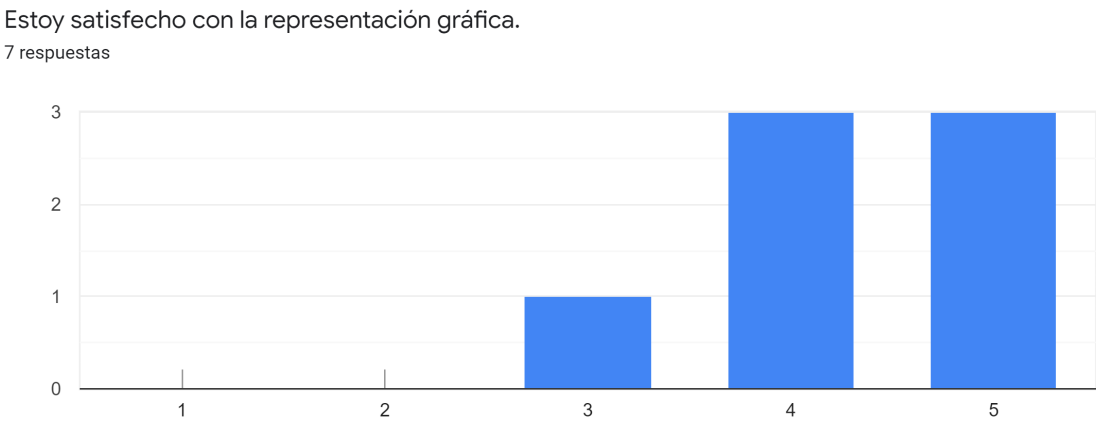

FIGURA D.12: Resumen de respuestas para la métrica «URLs frecuentes»

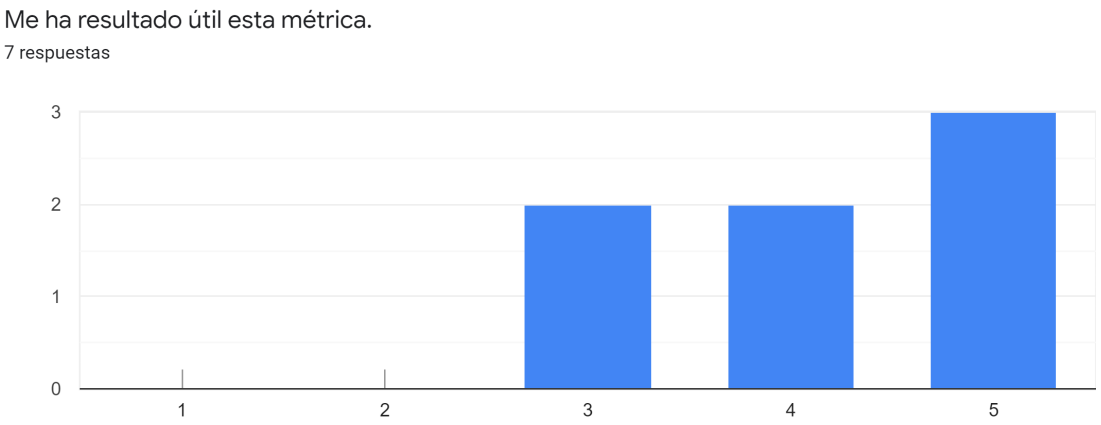

FIGURA D.13: Resumen de respuestas para la métrica «Análisis de sentimiento»

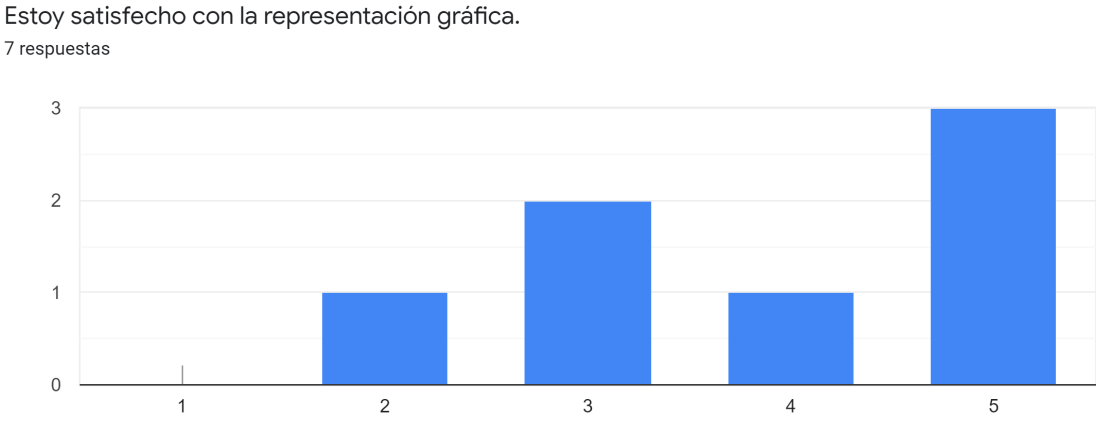

FIGURA D.14: Resumen de respuestas para la métrica «Análisis de sentimiento»

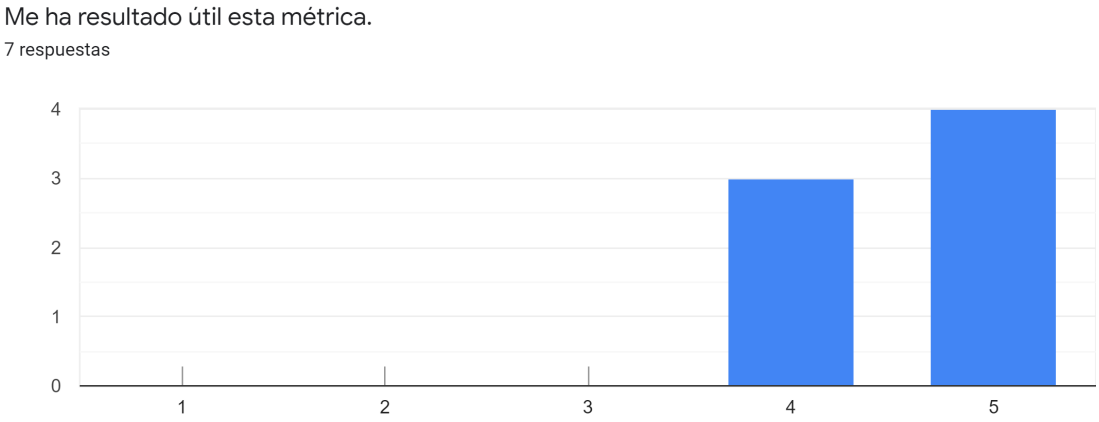

FIGURA D.15: Resumen de respuestas para la métrica «Análisis de emociones»

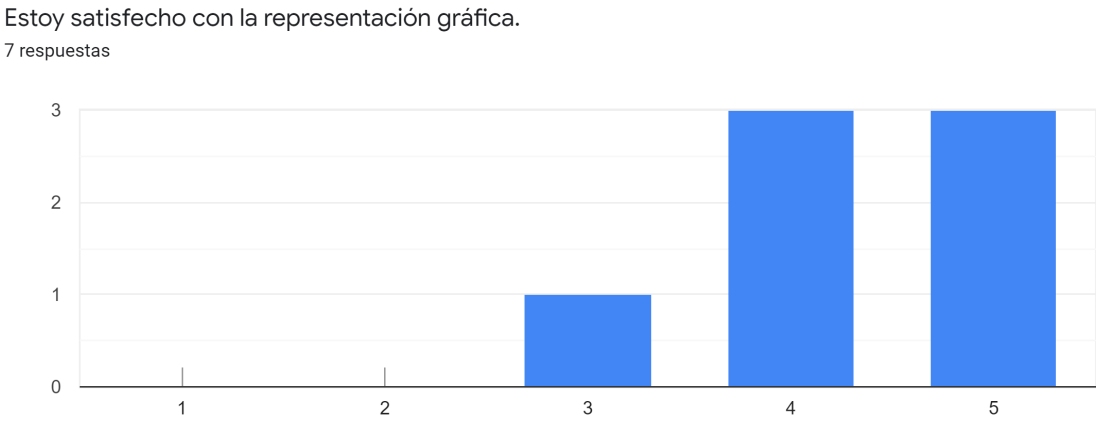

FIGURA D.16: Resumen de respuestas para la métrica «Análisis de emociones»

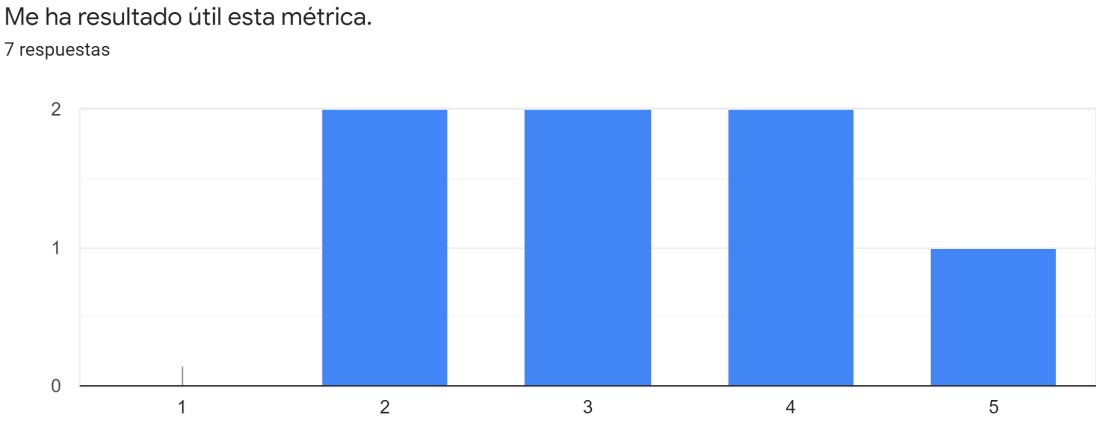

FIGURA D.17: Resumen de respuestas para la métrica «Distribución de fuentes»

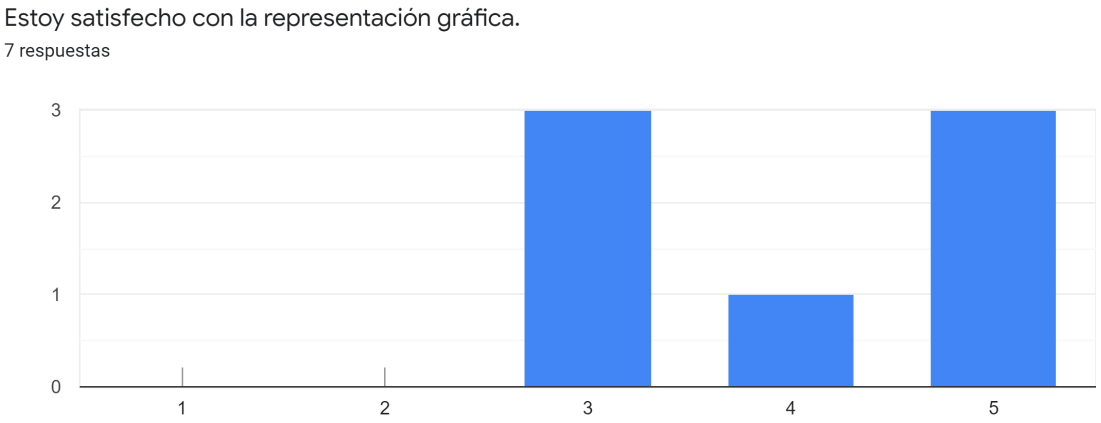

FIGURA D.18: Resumen de respuestas para la métrica «Distribución de fuentes»

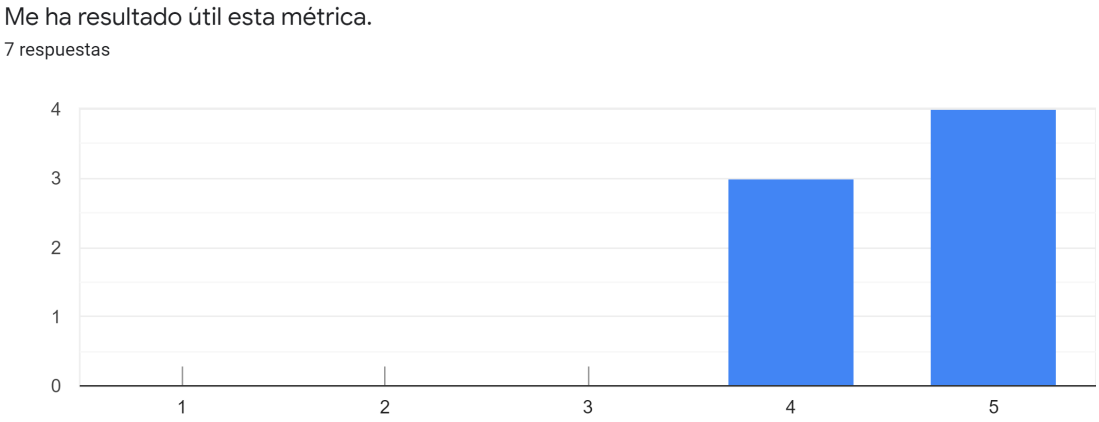

FIGURA D.19: Resumen de respuestas para la métrica «Localización de los usuarios»

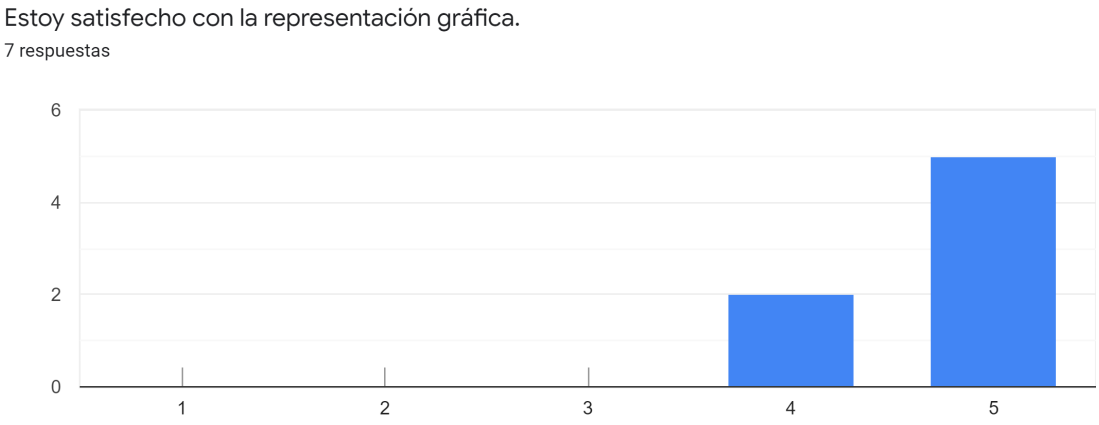

FIGURA D.20: Resumen de respuestas para la métrica «Localización de los usuarios»

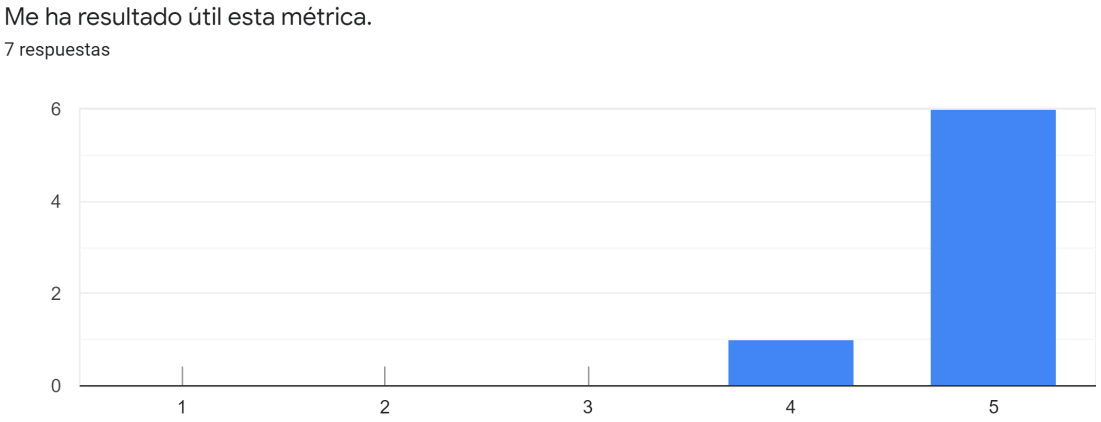

FIGURA D.21: Resumen de respuestas para la métrica «Tweets destacados»

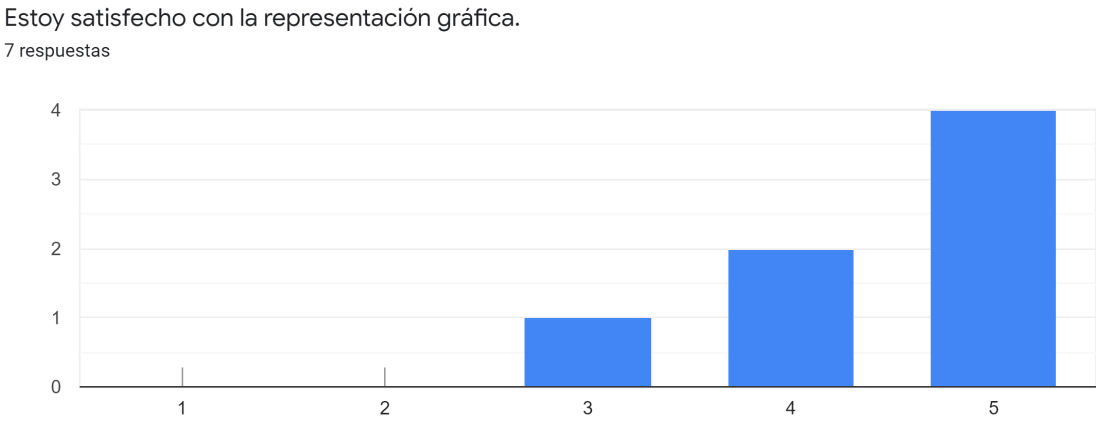

FIGURA D.22: Resumen de respuestas para la métrica «Tweets destacados»